2.24 WEAPON SELECTION

The purpose of the Weapon Selection Functional Element (WSFE) in Brawler is to model the pilot decision process of choosing a target and the best weapon to defeat it. Weapon-deploying platforms are limited to aircraft and surface to air missile (SAM) sites, target platforms are limited to aircraft, and weapons are limited to missiles and guns.

The WSFE is executed once during each pilot consciousness event. A consciousness event represents all of the pilot’s decision making, including the WSFE. During each consciousness event the WSFE will consider all known targets and all of the attacking pilot’s remaining weapons and select the highest scoring Weapon/Target Pair (WTP).

Many factors are involved in scoring the WTP. Scoring is first done based upon “hard” constraints that answer the question “Can I even get this shot off?”. If all of the hard constraints are passed, the next step is to compute a score for the WTP based upon the utility of the shot being considered. This score is a combination of a number of factors, such as the value of the target and the likelihood that the shot will succeed. The WTP pair achieving the highest score is the one selected.

Scoring is done by multiplying the value of the target by a surrogate probability of weapon kill, and then applying correction factors for additional effects. The surrogate probability of weapon kill (envelope level) is based on weapon envelope parameters and/or an empirical fit. The envelope parameters are sensitive to ranges, altitudes, speeds, target aspect angle, and attacker aim error. Because shots do not take place in a static environment, several correction factors are applied to determine the basic envelope level. These factors address specific considerations characterized by the following questions.

a. Will the range to the target be inside minimum range within the next few seconds? This is present primarily to encourage the pilot to select a shorter range weapon in high closure rate situations where the longer range weapon will no longer be best by the time the shot can be set up.

b. Is the target on the beam (i.e., is the target turned sideways to me, as opposed to nose-on or tail-on?) or turning onto the beam? This degrades the envelope for radar missile shots, anticipating or recognizing radar lock difficulties.

c. Is the target background ground or sky? This is used to degrade shots for semi-active radar or command guided missiles where the launcher's radar has no look-down/shoot-down capability.

d. Is it likely that single target track (STT) radar lock (when required) can be achieved and switches set by the time optimal shooting range will be achieved?

e. Will the attacker have to be occupied with illuminating the target? This factor is used to bias against radar missile shots, particularly semi-active radar missiles, and depends strongly on the anticipated illumination time requirement.

Additional corrections are applied to the basic envelope value, after multiplication by the target value. They consider hysteresis (the current target is preferred), a preference to target the assigned hostile in the flight's tactical plan, a preference to retain the current weapon choice, the degree to which the target is perceived as trying to directly engage the attacking pilot, whether other friendlies are interfering with the shot, and whether other friendlies have a significantly better shot at the same target. Finally, the user has an opportunity to influence the weapon/target pair selection through the Production Rules facility. Through Production Rules, a means of adding (or subtracting) value to (from) the score for any of the targets is available.

2.24.1 Functional Element Design Requirements

The WSFE will simulate the selection, or lack of selection, of a weapon and target by a human pilot based upon information legitimately available to that pilot. This FE will be a low-level decision function that contributes to the Pilot Posture decision process. Consideration of potential and desired weapon-target pairings will occur repeatedly so that transitions between those most practical for the current situation will occur over the course of the simulation.

a. Brawler will simulate the consideration of weapon-target pairing options by the pilot and use a scoring process to rank them into an order of preference.

b. Scoring of potential pairings will be a function of the value of a particular target, a surrogate probability of weapon kill, and additional correction factors for the situation. The probability of weapon kill (at the envelope level) will be based on envelope parameters or empirical data for the particular weapon. Envelop parameters should be sensitive to ranges, altitudes, speeds, target aspect angles, attacker aiming errors, and other applicable factors.

c. Brawler will simulate application of conditional factors that may override a specific pairing given certain conditions. Projection of target positions into the future, perceived target maneuvers, and factors affecting successful attack with a particular weapon will be considered in the process of refining scores for pairings.

d. Brawler will also allow for selection of weapon target pairings via user-defined (production) rules so that specific weapon attack conditions can be controlled during the course of the simulation.

2.24.2 Functional Element Design Approach

This section contains a description of the design approach that will implement the design requirements outlined in the previous section.

The weapon/target selection logic is organized in a “top-down” fashion for efficiency reasons. Tests that may disqualify a weapon or weapon/target combination from consideration are performed as early as possible in order to minimize the number of calculations required to reach a final selection decision. Selection of the weapon/target pair is performed by the following procedure.

1. Check launch-and-leave tactics. This may preclude weapon selection entirely.

2. Check weapon levels and number in air. This may eliminate a weapon type from consideration.

3. Score each weapon/target pair
    - 3.1 Determine if radar support is available
    - 3.2 Determine if seeker acquisition is possible
    - 3.3 Evaluate hardware fire control constraint tests
    - 3.4 Evaluate user-written fire control constraint tests
        - 3.4.1 Evaluate flight level constraints
        - 3.4.2 Evaluate constraints defined in production rules
    - 3.5 Compute weapon envelope level
        - 3.5.1 Compute basic envelope
        - 3.5.2 Adjust for additional factors
    - 3.6 Determine if interference from friendlies exists
4. Select weapon and target
    - 4.1 Determine weapon mode

Design Element 24-1: Check Launch-and-Leave Tactics

Subroutine SLWPN_LL checks to see if the aircraft is in the disengage phase of a launch and leave tactic. If in the disengage phase then the WTP selection process is unnecessary and is terminated. Launch and leave tactics can only be used with missiles that are not in command guided mode.

Design Element 24-2: Check Weapon Levels and Number in Air

Subroutine CHKWPN checks all of the weapon types that were initially on board the attacking platform to determine if any of them should not be considered for selection. A weapon type cannot be considered if: a) no weapons of that type remain; or b) the salvo limit for that weapon type has been reached, which means that the attacking platform has already launched and is currently supporting as many of that weapon type as it can.

Design Element 24-3: Score Each Weapon/Target Pair

Subroutine WPTGPR evaluates and scores the current WTP. WTP combinations that do not meet the minimum set of requirements are excluded from further consideration and receive a score of zero. These minimum requirements include: a) must not be out of ammunition for the weapon type; b) radar support must be available if the weapon requires it; c) if the missile seeker cannot be slaved to an avionics track, the pilot must have made a recent visual observation of the target; and lastly d) the target type is allowed for the weapon. If these criteria are satisfied, a basic score is assigned and then further adjusted for various effects. These effects include hysteresis for the current weapon and target, inducement to pursue the target designated by the flight leader, shot degradation when friendlies are in the way (risk of fratricide), shot devaluation when friendlies have better shots, enhancement of targets that are ahead of the attacking aircraft and threaten it, enhancement of targets or missiles as directed by productions rules, enhancement of targets designated by a GCI/AWACS controller. Several of the processes described here are encapsulated within lower level subroutines or functions. These are detailed below.

Design Element 24-4: Determine if Radar Support is Available

Logical function DSQUAL returns true if the current WTP requires radar support that is not available. This is an in-line function defined in subroutine *wptgpr*.

Design Element 24-5: Determine if Seeker Acquisition is Possible

Logical function CANTAQ returns true if the target cannot be acquired because the radar is unavailable and no recent visual observation of the target by the pilot has occurred. This is an in-line function defined in subroutine *wptgpr*.

Design Element 24-6: Hardware Fire Control Constraint Tests

Subroutine FCTEST performs hardware fire control tests on missiles. Guns do not undergo any of these hardware fire control tests. Missiles may be fired under different sets of conditions, referred to as launch modes. Each launch mode has its own preconditions for launching in that mode. For example, a missile might be fired in an undesignated mode, where all that is required is that the pilot have a recent visual observation of the target and the nose of the launching aircraft be pointed at the target. It might also be fired in a command guided mode, where guidance data will be transmitted from an avionics track on board the launching aircraft. It might also be fired in a passive seeker locked mode, where the missile seeker acquires the target signal before the missile is even fired. A single missile may have one or more launch modes. This means that each missile type under consideration has one or more sets of tests, at least one set of which must be passed in order for it to remain a candidate for selection. A bit mask that represents the test set will determine which of the tests in FCTEST are to be performed for one of the launch modes of the candidate missile. The complete set of tests and passing conditions is as follows:

a. seeker lock - must be locked on target

b. STT mode - current target must be the one being tracked

c. TWS track - track on target must be established

d. TWS or STT track - must be locked on target

e. IRST track - track on target must be established

f. post-stall condition- attacker cannot be in post-stall flight condition

g. recent visual - attacker must have recently seen target

h. RWR track - track on target must exist

i. target boresighted - attacker must be pointed at the target, to within the beamwidth of the candidate missile's seeker

j. ARM seeker - target's radar must be emitting in attacker's RWR band

k. ITB track - track on target must be established in the Integrated Track Bank

l. production rule range - target must be within range set by user

m. HMS FOV- target must be in attacker's Helmet Mounted Sight

FCTEST, then, will be called for each launch mode of a missile until a mode is found whose fire control tests can all be satisfied or until all of the possible launch modes have been tested.

Design Element 24-7: User-Written Fire Control Constraint Tests

In addition to the default fire control constraints that are embedded in the code or driven by missile-specific data, the user has the option of specifying Flight Level Fire Control Constraints and Production Rule Fire Control Constraints. These constraints can be written to either replace, further constrain, or relax the default conditions required to satisfy the conditions for selection of the WTP.

User influence over fire control is accomplished by giving the user access to a set of pre-defined logical and numeric variable names that can be used to construct boolean expressions that are then substituted for the default tests that would otherwise be applied for weapon selection or firing. The Flight Level Fire Control Constraints are associated with a flight of aircraft and are unchanged during the simulation. The Production Rule Fire Control Constraints facility extends the set of variables by allowing the user to actually define new variables that may be functions of other user defined variables or predefined variables. By employing the Production Rule Fire Control Constraints, the user can dynamically change the constraints during the simulation using the full power of the production rule syntax.

Specification of Flight Level and Production Rule constraints is done in the RULES file using the predefined variables. As an example of a Flight Level specification consider the following:

A user desires that a particular flight of aircraft select no targets at ranges greater than 10 miles under any circumstance. The following flight level fire control constraint would suffice for that flight of aircraft:

SELECT(STANDARD .AND. (RANGE_TO_TARGET < 10.))

This example is interpreted to require that all of the default constraints, denoted by “STANDARD”, be satisfied plus the additional range threshold constraint.

An example of a production rule constraint is as follows:

If, instead of associating the constraint with a flight of aircraft, we wanted the constraint applied only to aircraft 1 and 2 at 10 nmi and applied to aircraft 3 and 4 at 15 nmi, the “select” section of the rules would appear as:

```
SELECT
IF (IACID.EQ.1 .OR. IACID.EQ.2)
      SELECT(STANDARD .AND. (RANGE_TO_TARGET<10.) )
ELSEIF (IACID.EQ.3 .OR. IACID.EQ.4)
      SELECT(STANDARD .AND. (RANGE_TO_TARGET<15.) )
ENDIF
END END_FIRE_CONTROL_HANDLER
```

Subroutine FCSLCT performs the additional fire control tests based on flight level criteria and production rule criteria. The flight level and production rule constraints have precedence over the hardware constraints. So if the hardware fire control tests were passed but not the flight-level or production rule tests, the latter override the former and the current WTP is eliminated from consideration. The flight-level and production rule constraint tests are performed in order; hence the production rule constraints have precedence and therefore may override the flight-level constraints.

Design Element 24-8: Compute Weapon Envelope Level

The weapon envelope level computation is performed by two subroutines. Subroutine EVENVL calculates the basic level and subroutine ADJENV adjusts the level computed by EVENVL for additional factors. These subroutines are detailed below

Design Element 24-9: Compute Basic Level

Subroutine EVENVL computes the basic envelope level based on the position of the target relative to the weapon envelope, the probability of kill for the WTP, the target's value, and the number of weapons already targeted at or already en route to the target. Missile envelope range limits are determined from the projected positions of the attacker and target platforms so that overshoot of the target by the attacker may be anticipated. The gun range limits are determined from the current positions because of the much shorter flyout times for bullets. Other envelope parameters that are generated for both guns and missiles include the aimpoint position, steering error with respect to aimpoint, and the angle off, which is the angle between the line of sight to the target and the target's velocity vector.

Design Element 24-10: Adjust Basic Level

Subroutine ADJENV adjusts the envelope level for various factors including the anticipated degree of difficulty in acquiring radar lock, switchology delays, and the costs associated with illumination requirements. The ability to obtain lock is treated as a factor affecting the expected time delay until the shot can be fired. Switchology delays also contribute to this time delay. The time delay is used to compute an envelope degradation factor. This degradation factor is further adjusted to reflect constraints placed on the aircraft for the purpose of illuminating the target. The envelope level gets multiplied by the degradation factor to produce the adjusted envelope level.

Design Element 24-11: Determine if Interference from Friendlies Exists

Subroutine SET_HLDCOD determines if a target is ineligible because it is too near a friendly or a friendly has a better shot. A friendly "in the way" is determined on the basis of angular separation, range difference, and range rate difference. The determination of a friendly having a better shot is based on the range to an aimpoint and on the steering error.

Design Element 24-12: Select Weapon and Target

Subroutine SELWPF selects the highest scoring WTP from the envelope level list generated in WPTGPR. If the highest score is below a minimum acceptable value, then no WTP is selected. The time at which the WTP was selected and the time at which the weapon can be fired are set. A call to subroutine MISLMD (detailed below) recomputes the envelope level as a function of the weapon mode. Finally the time at which switch setting starts is adjusted depending on the weapon mode.

Design Element 24-13: Determine Weapon Mode

Subroutine MISLMD determines the mode of the selected weapon. The possible modes are mode 0 (pilot not interested in firing), mode 1 (pilot interested in firing), and mode 2 (shot is imminent). The mode is initially set to zero. If the target will be in range within 10 seconds and the steering error will be within limits in 10 seconds then the mode is increased to 1. Mode is increased to 2 if target ID requirements are satisfied, the target is not already saturated with missiles, the target is or will be within weapon envelope within 2 seconds, the attacking pilot believes the target is within range, the attacking pilot perceives no risk of fratricide, and the attacking pilot perceives that no friendly has a better shot. An additional mode bit (bit 3) is set to indicate if one or more weapons are already in the air.

2.24.3 Functional Element Software Design

This section contains the software design necessary to implement the functional element requirements and the design approach defined in the preceding sections. The first subsection describes the subroutine hierarchy and describes how the subroutines work to select the WTP. Subsequent subsections contain functional flow diagrams and describe all important operations represented by each block in the diagrams.

Weapon Selection Subroutine Hierarchy and Description

Weapon selection is performed by subroutine *selwpn*. The major routines called by the weapon selection scoring algorithm and their purpose are given below with the indentation of the routine name used to indicate the level of the routine within the calling tree.

- *selwpn* - weapon/target pair selection executive
  - *slwpn_ll* - check launch-and-leave tactics
  - *chkwpn* - check weapon levels and number in air
  - *wptgpr* - score each weapon/target pair
    - *dsqual* - determine if radar support is available
    - *cantaq* - determine if seeker acquisition is possible
    - *fctest* - hardware fire control constraint tests
    - *fcslct* - user-written fire control constraint tests
    - *evenvl* - compute basic envelope level
    - *adjenv* - adjust envelope level for acquisition delays
    - *set_hldcod* - determine if interference from friendlies exists
  - *selwpf* - select weapon and target
    - *mislmd* - determine weapon (missile) mode

A number of secondary and utility subroutines are used in the weapon selection process as described below.

aimpt	Calculates a rudimentary aim point against a target.
armacq	Determines if ARM seeker has acquired its target.
asstgt	Returns the aircraft that the conscious pilot has been assigned to attack.
avfrat	Evaluates possibility of fratricide with particular weapon/target pair and a particular friendly aircraft.
envlvg	Returns a normalized envelope score for a gun/target pair.
envlp1	Calculates min and max weapon range using standard algorithm.
envlp2	Calculates min and max weapon range using F-15 HUD algorithm.
envlp3	Calculates min and max weapon range using multidimensional tables.
envlvl	Returns a normalized envelope score for a missile/target pair.
gainar	Calculates gain of target emitter in direction of ARM seeker.
ggund	Loads gun characteristics data.
gmisld	Loads missile characteristics data.
gunenv	Similar in function to *mslenv* for gun; returns attributes of envelope.
gunpar	Returns gun tracking data.
lockid	Determines if radar has a track against a particular target.
mslenv	Executive directing computation of missile envelope variables: minimum range, maximum range, aim point, steering error, aspect, angle above horizon, etc.
mystor	Fetches remaining weapons stores levels.
selwpi	Initializes variables needed for the weapon/target selection.
snarm	Calculates signal to noise ratio at ARM seeker.

Input and output arguments to each routine are documented in the subroutine template at the beginning of each routine. Figure 2.24-1 presents this information in standard calling-tree format.

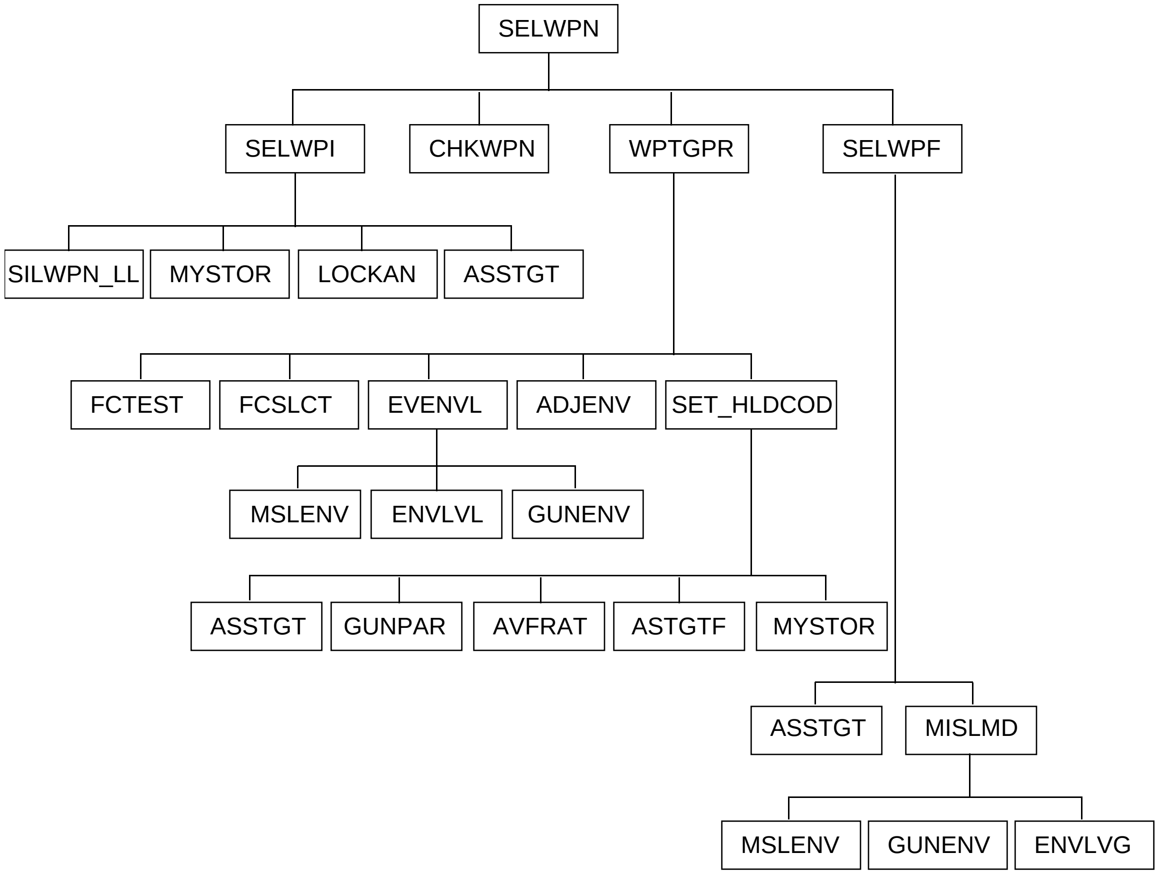

FIGURE 2.24-1. Weapon/Target Selection Calling Tree.

The principal data structures (common blocks) involved in the weapon selection process are described below.

/*envdat*/ Missile envelope characteristics data.

/*extst*/ External status common block. Contains “ground truth information about all of the players in the scenario.

/*fcstat*/ Fire control status. Holds information related to weapon selection and status. *mslpp* identifies the fox number of the selected missile; *minair* reflects the number of missiles in the air for each fox number; and *mxmair* indicates the maximum number of missiles in the air for each fox number.

/*mind2*/ Pilot mental model common. *mxtgtd* reflects firing doctrine with regard to the maximum number of weapons to simultaneously target at one aircraft, while *rpeak* indicates doctrine with respect to how deep into the envelope (in range) one should go before shooting. These are set for each flight in the engagement.

/*mind3*/ Pilot mental model common. Contains pilot perceptions of state variables such as position, velocity, acceleration of all aircraft of which the pilot is aware.

/*mind4*/ Pilot mental model common. Target value information is used to obtain pair utilities, and “intent” and other higher level situational assessment variables contribute to the calculation of additional variables that modify the basic goodness of shot envelope level.

/*misdat*/ General missile characteristics; type of seeker, type of guidance, etc.

/*ppost*/ The variables that begin with “ppm” are the output variables of the weapon/target decision.

/*prjct*/ The projected target state vector is used by *evenvl* to get envelope levels based on projected situations.

/*rdrsta*/ Radar state information needed to assess radar conflicts and ability to lock in addition to presence of aircraft tracks in track bank.

Each included file has an associated documentation file, named xxx.inc.doc, where xxx is the name of the include file. The documentation files contain the descriptions of all of the parameters and variables in the include file.

Subroutine *selwpn* operates by considering weapon and target choice to be a unified decision; that is, a value is placed on each weapon/target <u>pair</u> as opposed to sequential choices of target and weapon. This decision is performed as part of the pilot posture decision. For each candidate weapon/target pair, *selwpn* generates a value for the selection of the pair. Some weapon/target pairs may not be considered, due to such factors as out-of-munitions, too many missiles already in the air (see /*misdat*/*mxnair*), target missing from the high priority target list, radar conflict (i.e., an STT radar is already in use for guiding a radar missile at another target), or no RWR track exists (for an ARM missile).

Initial weapon/target values are obtained by calls to *evenvl*, which in turn calls either *mslenv* or *gunenv* to determine aerodynamic envelope parameters, such as *rmin* and *rmax*. The *envlvl* subroutine is then called to evaluate the desirability of the shot, based on the situation now and how it will develop over the next 5 seconds. The result of this calculation is the normalized (0-1) envelope level.

At this point other factors that reflect the total situation must be considered to arrive at the true value of shooting that weapon at that target. *evenvl* begins this process by multiplying the envelope level by a factor that accounts for other missiles in the air against the target, the utility of engaging the hostile (the score for killing the target), and a rough estimate of the weapon's reliability and lethality. Upon returning to *selwpn*, a call to *adjenv* further refines the value by considering factors relating to the ability to achieve and maintain lock (knowledge of target elevation, if not locked, and relation of target to the Doppler notch). *adjenv* also applies penalties for shots requiring radar illumination to reflect the effect of being "tied up" by this requirement. This penalty is a function of the estimated time-of-flight of the missile.

selwpn applies additional adjustments to the value to reflect hysteresis (i.e., a preference for the current weapon and current target), tactical assignments, the degree of threat to the shooter presented by the target, interference by other friendlies, and other friendlies with a better shot. An additional penalty is applied for the repeated selection of a weapon requiring seeker lock without that weapon having achieved lock on the target. Finally, adjustments may be made to the score if the user has specified, through production rule variables *prtgtd* and *prmisl*, that he wishes to bias the decision towards a particular set of targets or a particular weapon type.

Once a selection is made (that is, the highest valued weapon/target pair has been determined), a selection of the missile mode (*mslmd*) is made through a call to *mislmd*. A value of 0 indicates that no imminent weapon firing is anticipated, 1 indicates that a shot is imminent but that the pilot is not yet ready to fire, and 2 indicates that the shot can be made now (but it may be advisable to wait and try to improve the shot). This categorization is determined by checking through two sets of conditions. All of one group of conditions must be met before mode 1 is enabled; in addition, all of a second group must be met before mode 2 is enabled. The missile mode not only determines ability to fire, it determines the balance between weapon-aim and general pursuit values used in the aircraft maneuver decision. Weapon aiming (pointing the nose at a computed aimpoint and keeping it there) forms the total offensive value component while in missile mode 2; a mixture of weapon aiming and values designed to improve and maintain position are used in missile mode 1; and missile mode 0 uses only pursuit values.

Subroutine SELWPN

Subroutine SELWPN selects the best weapon/target combination out of all possible combinations. It is the executive subroutine of the WSFE. Figure 2.24-2 is the functional flow diagram that describes the logic used to implement SELWPN. The blocks are numbered for ease of reference in the following discussion.

Block 1: Subroutine SELWPI is called to initialize variables needed for the WTP selection decision. SELWPI also contains the call to subroutine SLWPN_LL.

Block 2: This block represents the call to SLWPN_LL (detailed below), determining the missile (or weapon) mode, and the determination of the existence of hostile aircraft.

Block 3: Decision to bypass weapon selection based on results from Block 2.

Block 4: Additional initialization is performed here, including evaluation of radar status, assessment of rear quarter threats, and recording target assignments and what types of weapons are currently available.

Blocks 5 and 6 are in a loop over all of the possible weapon types.

Block 5: This block represents the call to subroutine CHKWPN which is detailed below.

Block 6 is in a loop over all of the known target aircraft.

Block 6: This block represents the call to subroutine WPTGPR which is detailed below.

Block 7: This block represents the call to subroutine SELWPF which is detailed below.

Block 8: Test whether a WTP was selected.

Block 9: The additive drag/signature is set to off for the attacking aircraft.

Block 10: Test whether Brawler is operating in confederation with one or more other simulations.

Block 11: A status update message protocol data unit (PDU) is sent by a call to subroutine C2_MESSAGE if the target has changed.

Block 12: This block represents the call to subroutine SELWPF_RAIL that creates or destroys missiles on the rail as required.

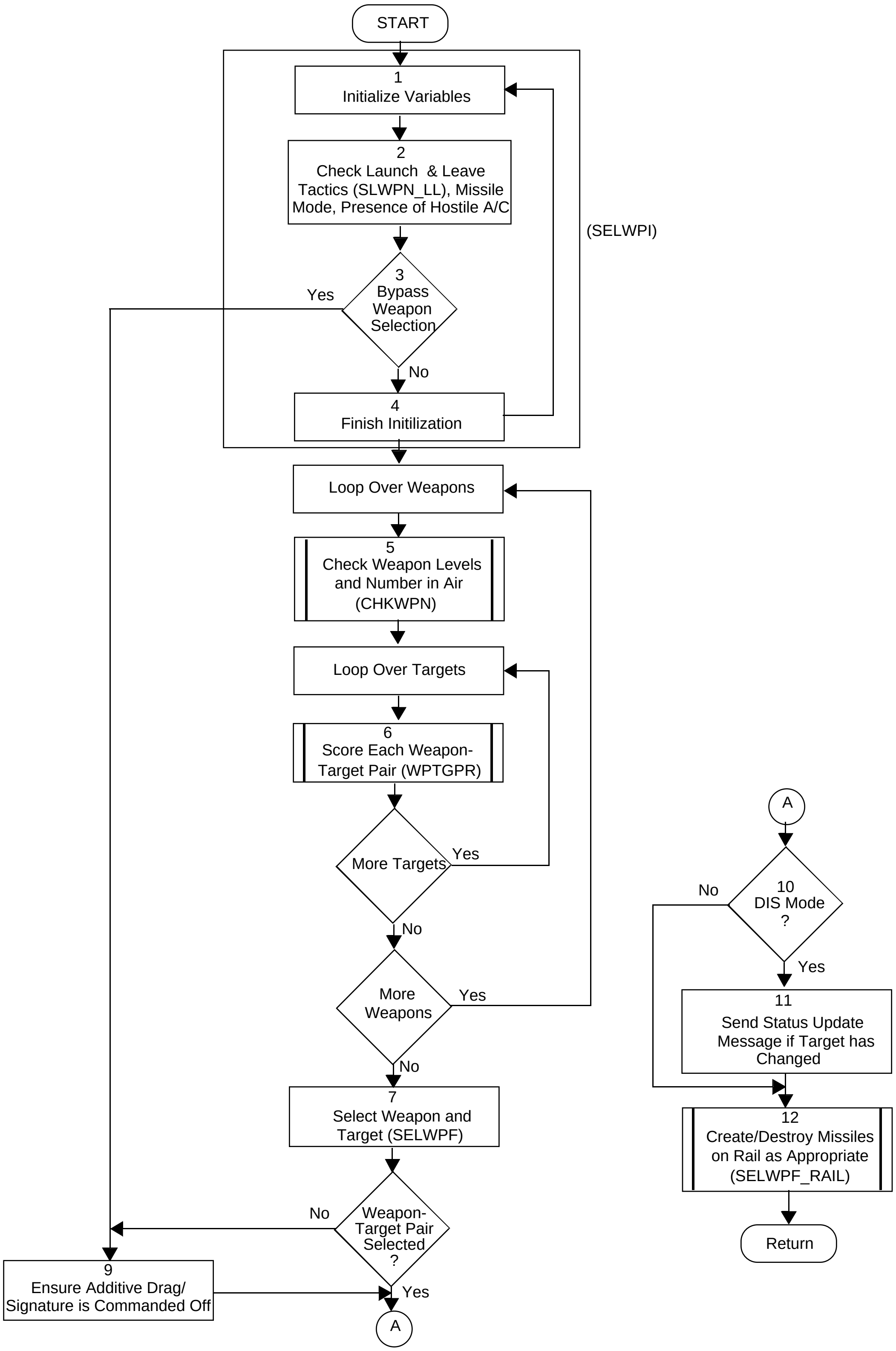

FIGURE 2.24-2. SELWPN Functional Flow Diagram.

Subroutine SLWPN_LL

Subroutine SLWPN_LL determines if a launch-and-leave tactic is currently being employed by the attacking aircraft. Figure 2.24-3 is the functional flow diagram that describes the logic used to implement SLWPN_LL. The blocks are numbered for ease of reference in the following discussion.

Block 1: Subroutine INDUPK is called to get parameters that collectively determine launch-and-leave status and a test is performed on the status parameters to determine whether or not the aircraft is in the disengage phase of a launch and leave tactic.

Block 2: Decision based on whether launch-and-leave is in effect and whether the pilot is about to use an IR missile.

Block 3: The ‘noslct’ flag is set true which tells the calling routine SELWPI to bypass the weapon/target selection decision with no WTP selected.

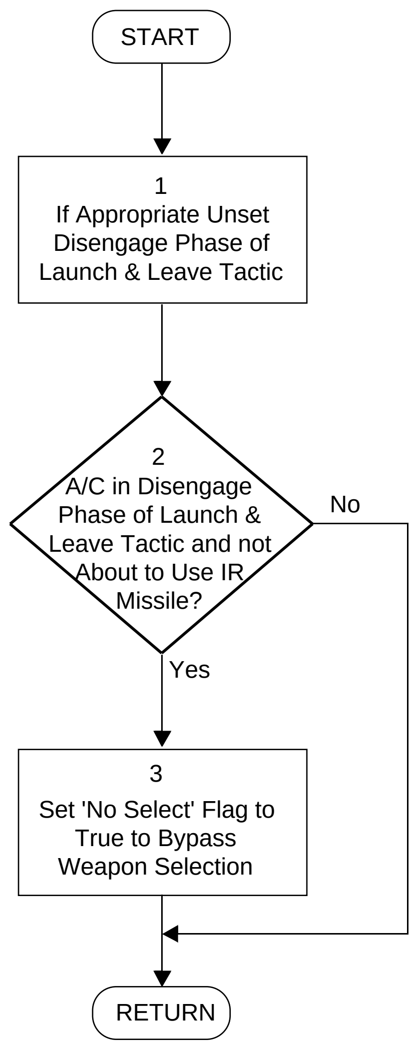

FIGURE 2.24-3. SLWPN_LL Functional Flow Diagram.

Subroutine CHKWPN

Subroutine CHKWPN determines if weapons of the type being considered are available and the number of weapons that have been fired that are in the air. Figure 2.24-4 is the functional flow diagram that describes the logic used to implement CHKWPN. The blocks are numbered for ease of reference in the following discussion.

Block 1: Test of whether any weapons of the particular type were initially stored on-board.

Block 2: Test of whether any weapons of the particular type remain (have not all been fired).

Block 3: Test of type of weapon being considered - gun or missile.

Block 4: Determines if the maximum number of missiles in the air has been reached or exceeded. Sets variable ‘mxguid’ appropriately.

Block 5: Sets mxguid to false for guns.

Block 6: Test of whether the number of weapons currently in the air equals or exceeds the maximum allowed.

Block 7: Statement that the weapon is available for selection. Implemented by the ‘skpwpn’ (read “skip weapon”) logical variable = false.

Block 8: Statement that the weapon is not available for selection. Implemented by the ‘skpwpn’ logical variable = true.

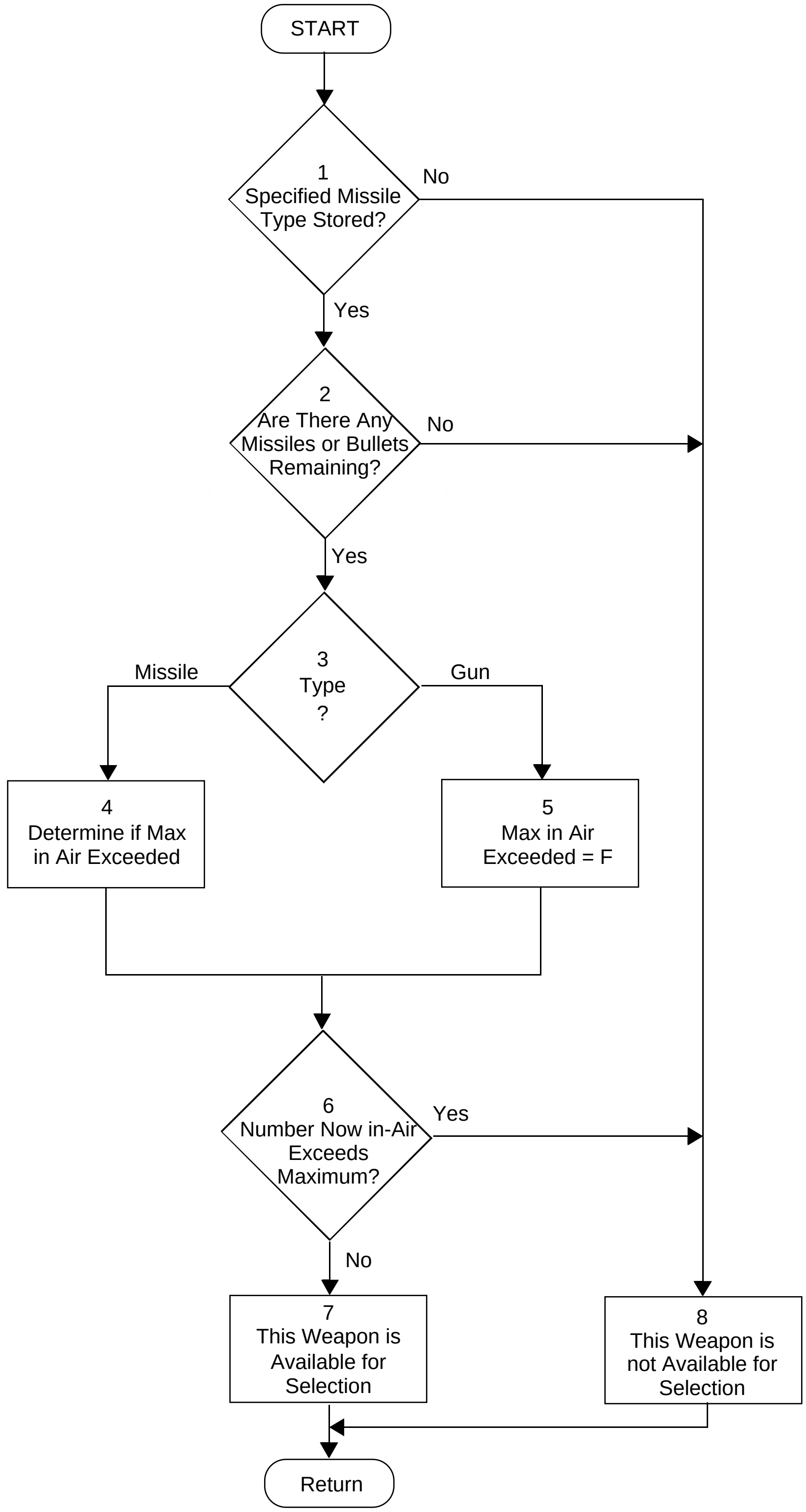

FIGURE 2.24-4. CHKWPN Functional Flow Diagram.

Subroutine WPTGPR

Subroutine WPTGPR is the executive routine for assigning a score to the WTP. It uses several routines described below to give the WTP a score ranging from 0 to 1. Figure 2.24-5 is the functional flow diagram that describes the logic used to implement WPTGPR. The blocks are numbered for ease of reference in the following discussion.

Block 1: Test whether the weapon type is compatible with the target type.

Block 2: Test if target is the same side as the attacker and fratricide has been prohibited by the user.

Delete this block and renumber the remaining blocks.

Block 3: Test whether radar support is available using function statement DSQUAL detailed below.

Block 4: Test whether seeker acquisition is possible using function statement CANTAQ detailed below.

Block 5: Test whether hardware fire control constraints are met using subroutine FCTEST detailed below.

Block 6: Logical variable ‘phrdwr’ = true indicating passed hardware fire control.

Block 7: Logical variable ‘phrdwr’ = false indicating failed hardware fire control.

Block 8: Test whether user-written fire control constraints are met using subroutine FCSLCT detailed below.

Block 9: Calculation of the basic weapon envelope level using subroutine EVENVL detailed below.

Block 10: Adjustment of the envelope level using subroutine ADJENV detailed below.

Block 11: Test if the current target is the same as the target that was selected last time.

Block 12: Envelope level is increased if number of missiles already targeted on this target does not equal or exceed the maximum number allowed.

Block 13: Test if the current weapon is the same as the weapon that was selected last time.

Block 14: Envelope level is increased if number of missiles already targeted on this target does not equal or exceed the maximum number allowed.

Block 15: Envelope level is decreased if seeker acquisition is required.

Block 16: Test whether the current target is the target assigned by the flight leader.

Block 17: Envelope level is increased number of missiles already targeted on this target does not equal or exceed the maximum number allowed.

Block 18: Test whether other friendly aircraft exist.

Block 19: Test whether a friendly aircraft is in the way of the shot or has a better shot. Test performed via a call to subroutine SET_HLDCOD detailed below.

Block 20: Envelope level is decreased.

Block 21: Test whether the attacker is a SAM site or an aircraft.

Block 22: Test whether there is a hostile aircraft ahead of the attacking aircraft.

Why is there no reference to the call to wtsem? I didn't see it. It was buried on the right side of a complex statement. Its existence as a function was not obvious. It is now recognized and explained in the block 24 description below.

Block 23: Envelope level is increased as a function of the Self-Engagement Measure (function wtsem, a metric for determining who is a threat to whom) of the hostile aircraft ahead.

Block 24: Test whether a production rule target bias value exists.

Block 25: Envelope level is adjusted by the production rule target bias value.

Block 26: Test whether a production rule missile bias value exists.

Block 27: Envelope level is adjusted by the production rule missile bias value.

Block 28: Test whether a GCI vectoring bias value exists.

Block 29: Envelope level is adjusted by the GCI vectoring bias value.

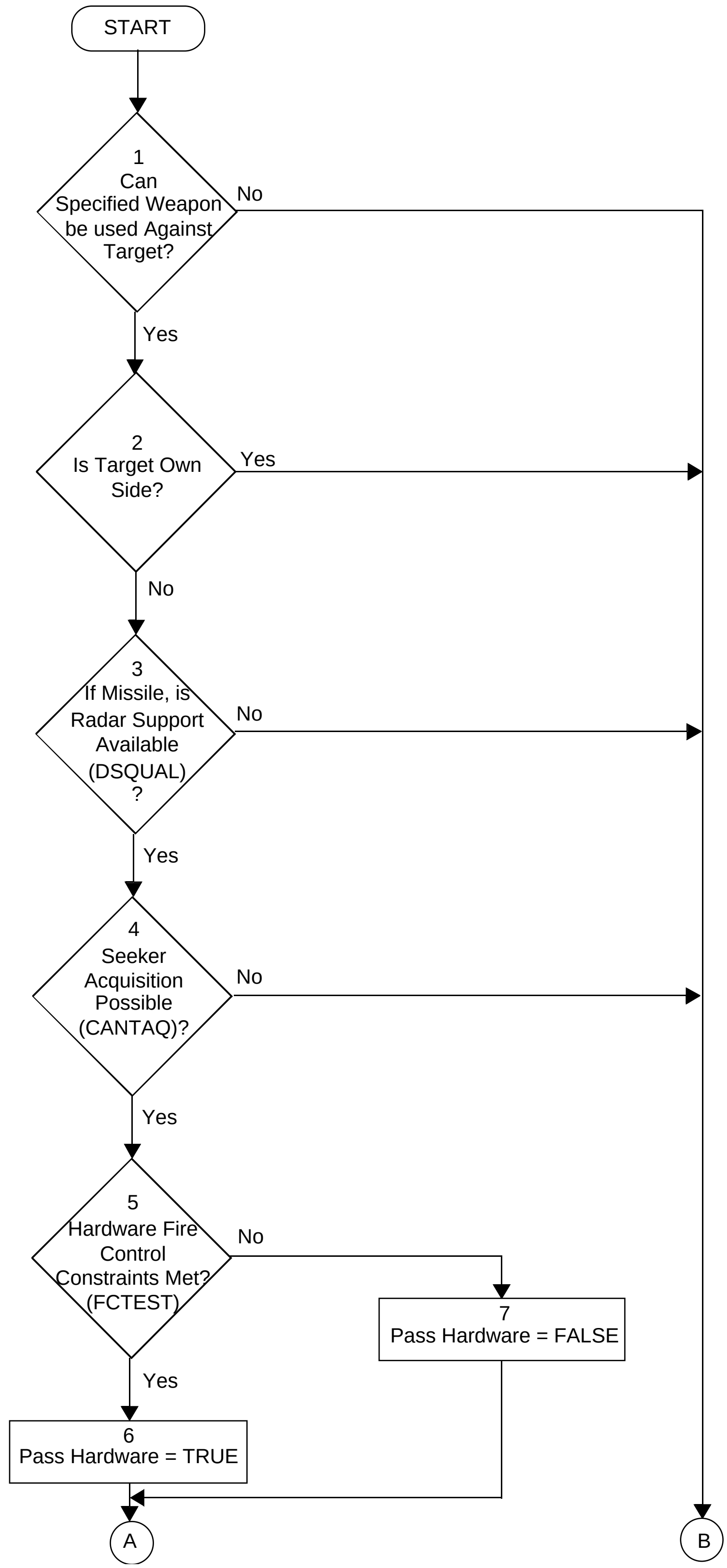

FIGURE 2.24-5. WPTGPR Functional Flow Diagram (Page 1 of 3).

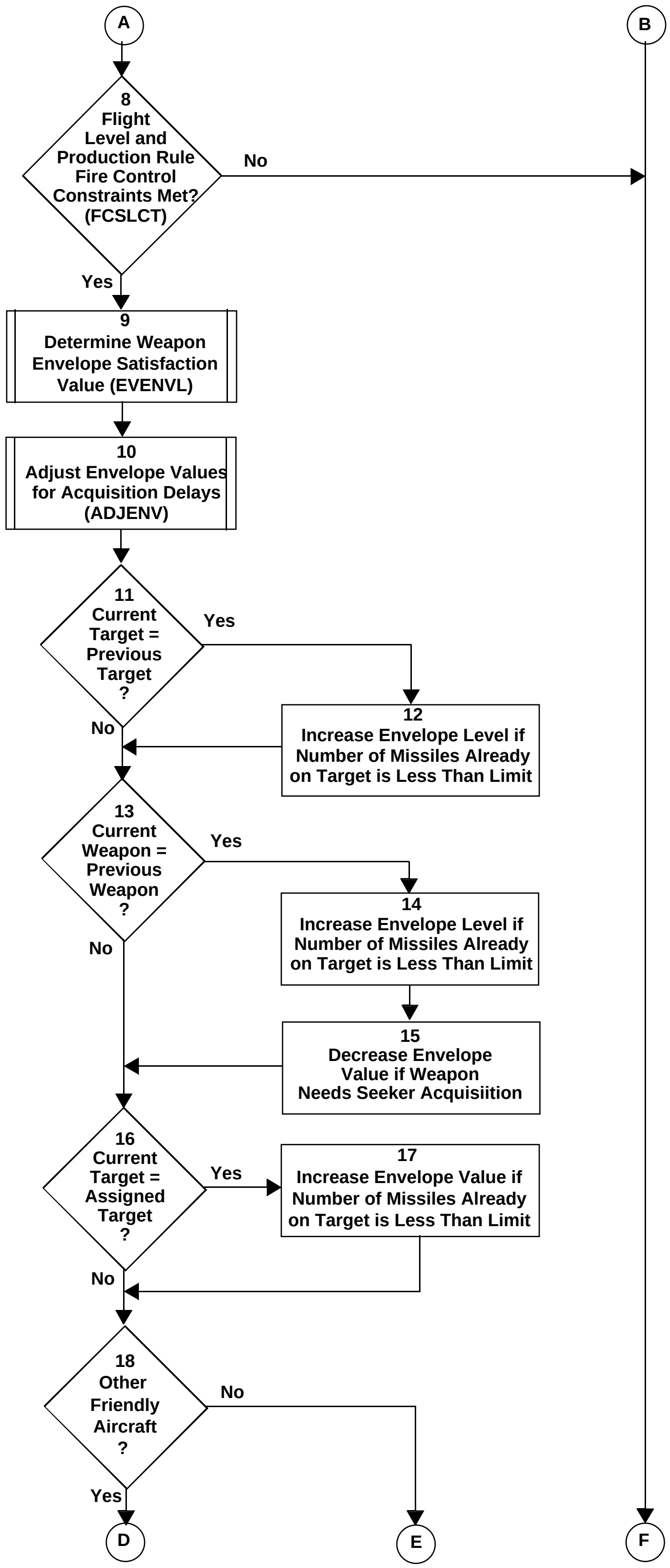

FIGURE 2.24-5. WPTGPR Functional Flow Diagram (Page 2 of 3).

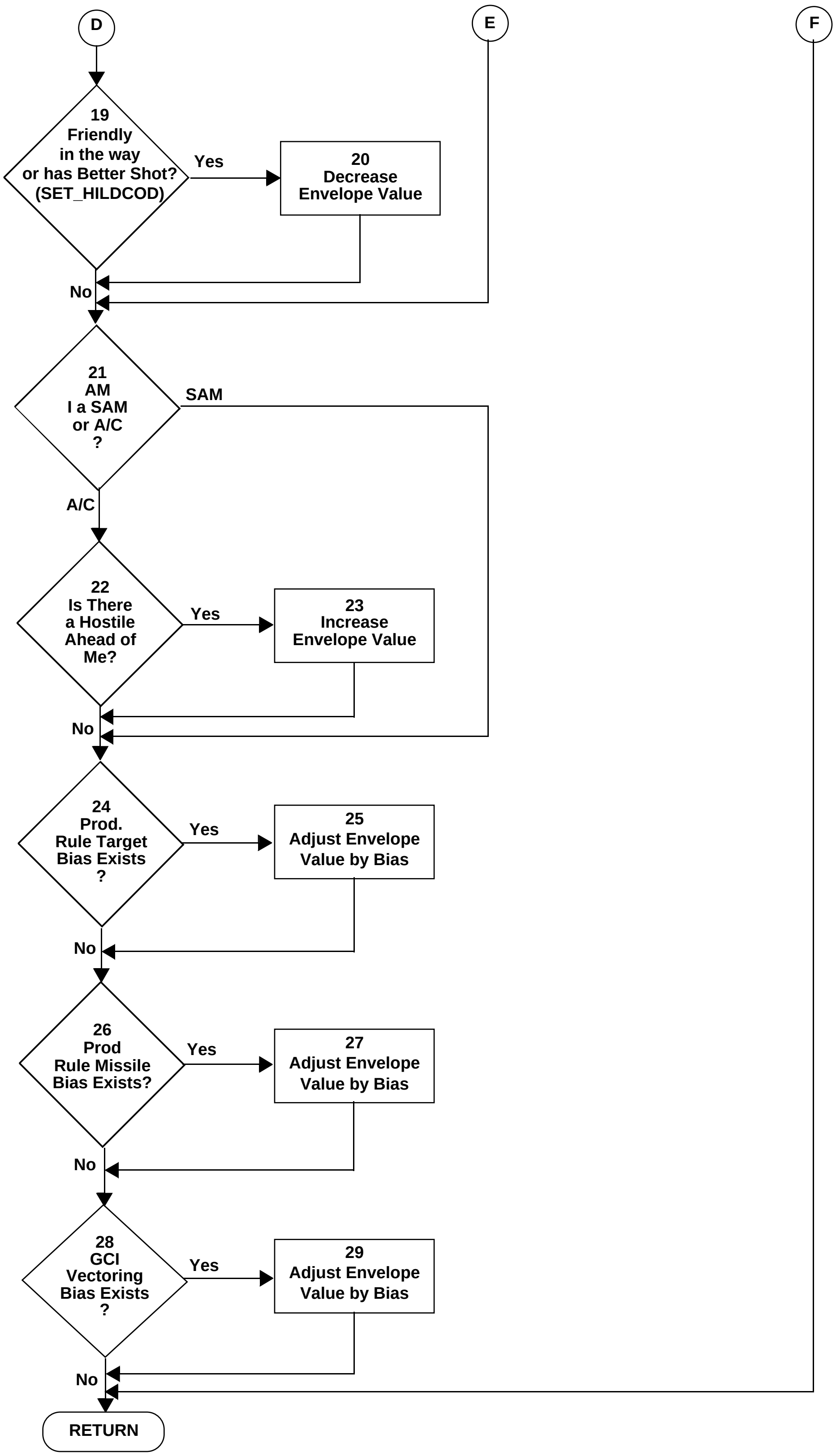

FIGURE 2.24-5. WPTGPR Functional Flow Diagram (Page 3 of 3).

Function Statement DSQUAL

Function statement DSQUAL determines if radar support is available for a missile that requires illumination. Figure 2.24-6 is the functional flow diagram that describes the logic used to implement DSQUAL. The blocks are numbered for ease of reference in the following discussion.

Block 1: Test whether there are any missiles already in the air requiring illumination by the attacking aircraft. This test is based upon pilot perception, not on ground truth.

Block 2: Test whether the on-board radar has a track bank. This capability determines whether or not the radar can support more than one missile at a time.

Block 3: Test whether the current target is that of the first missile that requires illumination.

Block 4: Test whether the missile has a semi-active seeker.

Block 5: Test whether the missile can be launched in command guided mode.

Block 6: Function statement DSQUAL is set to true.

Block 7: Function statement DSQUAL is set to false.

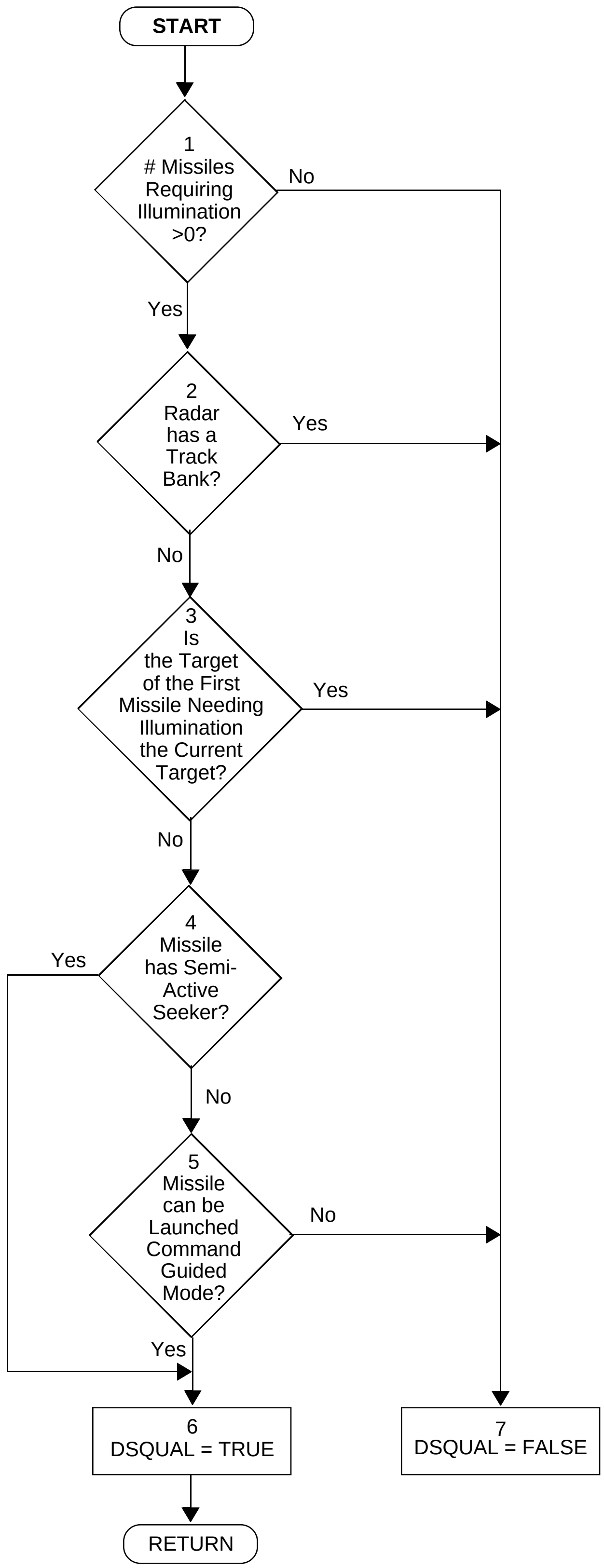

FIGURE 2.24-6. DSQUAL Functional Flow Diagram.

Function Statement CANTAQ

Function statement CANTAQ determines if seeker acquisition is possible. Figure 2.24-7 is the functional flow diagram that describes the logic used to implement CANTAQ. The blocks are numbered for ease of reference in the following discussion.

Block 1: Test whether there are any missiles already in the air requiring illumination by the attacking aircraft. This test is based upon pilot perception, not ground truth.

Block 2: Test whether the target has been detected by means other than visual.

Block 3: Test whether the radar is in STT mode or a track is established on the selected target.

Block 4: Function statement CANTAQ is set to true.

Block 5: Function statement CANTAQ is set to false.

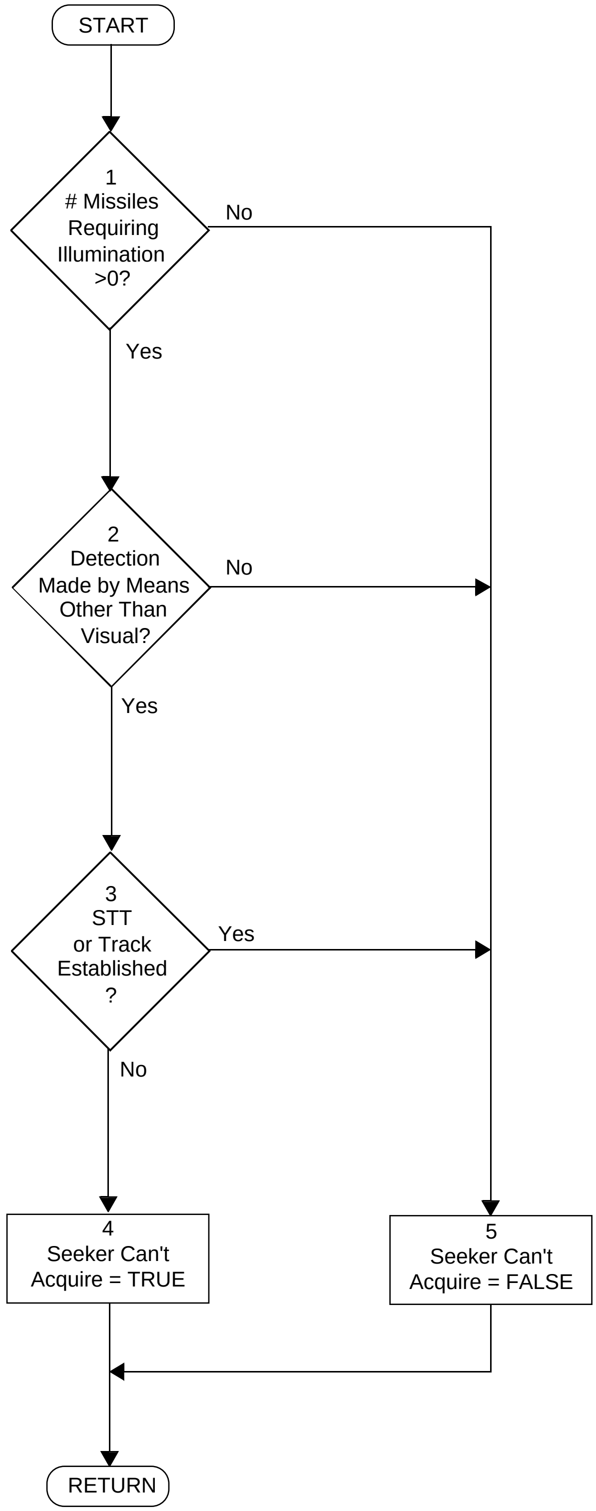

FIGURE 2.24-7. CANTAQ Functional Flow Diagram.

Subroutine FCTEST

Subroutine FCTEST determines if hardware fire control constraints are satisfied for the current WTP. Figure 2.24-8 is the functional flow diagram that describes the logic used to implement FCTEST. The blocks are numbered for ease of reference in the following discussion.

Block 1: Test whether weapon is a gun or missile.

Block 2: If the weapon is a gun then fire control is passed (logical variable 'pass' is set to true) and the routine returns. Selection of a gun is not subject to hardware fire control constraints.

Blocks 4 through 61 are in a loop over all of the possible fire control tests.

Block 3: Test whether the current indexed test is required.

Block 4: This block represents a case statement type selection mechanism for a particular test according to the test index 'itst'. This case statement structure is implemented in the code with a large 'if - elseif' structure.

Block 5: This is the first block in the seeker lock test. Test whether the current WTP is the previously selected WTP.

Block 6: If the WTP is the previous one then fire control is failed (logical variable 'pass' is set to false).

Block 7: Test whether the missile has an IR seeker.

Block 8: Test whether the missile has a semi-active seeker.

Block 9: Test whether the IR seeker has acquired the target. If yes then seeker lock test passes.

Block 10: Test whether there is only one seeker. If so then the routine aborts because this is an illegal launch condition for a missile with only a semi-active seeker.

Block 11: Test whether the missile has an active seeker.

Block 12: Test whether the missile has an ARM seeker. If not then the routine aborts since the missile has no seeker or has a seeker of an unknown type.

Block 13: Test whether there is only one seeker. If so then the routine aborts since this is an illegal launch condition for a missile with only an active seeker.

Block 14: Test whether the ARM seeker acquired the target. If yes then seeker lock test passes.

Block 15: If this block is reached then fire control has failed (logical variable 'pass' is set to false) and the routine returns.

Blocks 16 and 17 are in a loop over the number of radar antennas on board the attacking platform.

Block 16: This is the first block in the STT mode test. Test whether the current antenna is in STT mode.

Block 17: Test whether the antenna is locked on the selected target. If yes then the STT mode test passes.

Block 18: If this block is reached then fire control has failed (logical variable 'pass' is set to false) and the routine returns.

Block 19: This is the first block in the TWS track test. Test whether a Sensor Fusion Device (SFD) exists.

Block 20: Subroutine HAVTRK is employed to retrieve a list of current TWS tracks on the selected target.

Block 21: Test whether a TWS mode track on the selected target has been established. If yes then the TWS track test passes.

Block 22: If this block is reached then fire control has failed (logical variable 'pass' is set to false) and the routine returns.

Block 23: This is the first block in the TWS or STT track test. Decision on whether a Sensor Fusion Device (SFD) exists.

Block 24: Subroutine LOCKID is employed to retrieve radar TWS and STT mode status data.

Block 25: Test whether an STT lock or a TWS mode track on the selected target has been established. If yes then the TWS or STT track test passes.

Block 26: If this block is reached then fire control has failed (logical variable 'pass' is set to false) and the routine returns.

Block 27: This is the first block in the IRST track test. Decision on whether a Sensor Fusion Device (SFD) exists.

Block 28: Subroutine HAVTRK is employed to retrieve IRST track data on the selected target.

Block 29: Test whether an IRST track on the selected target has been established. If yes then the IRST track test passes.

Block 30: If this block is reached then fire control has failed (logical variable 'pass' is set to false) and the routine returns.

Block 31: This is the first block in the post-stall condition test. Test whether the attacking aircraft's angle of attack is past the limit at which this type of missile can be launched. If no then the post-stall condition test passes.

Block 32: If this block is reached then fire control has failed (logical variable ‘pass’ is set to false) and the routine returns.

Block 33: This is the first block in the recent visual test. Test whether the amount of time since the attacking aircraft’s pilot last saw the target exceeds the maximum allowed amount of time. If no then the recent visual test passes.

Block 34: If this block is reached then fire control has failed (logical variable ‘pass’ is set to false) and the routine returns.

Block 35: This is the first block in the RWR track test. Test whether a Sensor Fusion Device (SFD) exists.

Block 36: Subroutine HAVTRK is employed to retrieve RWR track data on the selected target.

Block 37: Test whether an RWR track on the selected target. If yes then the RWR track test passes.

Block 38: If this block is reached then fire control has failed (logical variable ‘pass’ is set to false) and the routine returns.

Block 39: This is the first block in the target in FOV test. Decision on whether the target is within the angular FOV of the attacking aircraft’s radar antenna. If yes then the target in FOV test passed.

Block 40: If this block is reached then fire control has failed (logical variable ‘pass’ is set to false) and the routine returns.

Block 41: This is the first block in the ARM seeker test. Subroutine HAVSKR is employed to retrieve missile seeker data.

Block 42: Test whether the missile has an ARM seeker.

Block 43: Test whether attacking aircraft has an RWR.

Block 44: Test whether missile has only one seeker. If yes, an abort occurs, because this test is only applied to missiles with ARM seekers. If no then the ARM seeker test passes.

Block 45: Test whether the RWR is on.

Block 46: Test whether the RWR is tracking the target.

Block 47: Test whether the target’s radar is emitting at a wavelength that can be detected by the ARM seeker. If yes then the ARM seeker test passes.

Block 48: If this block is reached then fire control has failed (logical variable ‘pass’ is set to false) and the routine returns.

Block 49: This is the first block in the ITB (Integrated Track Bank) track test. Subroutine HAVTRK is employed to retrieve SFD track data on the selected target.

Block 50: Test whether an SFD track is established. If yes then the ITB track test passes.

Block 51: If this block is reached then fire control has failed (logical variable 'pass' is set to false) and the routine returns.

Block 52: This is the first block in the production rule range test. The range to the target is calculated.

Block 53: Test whether the target is within the production rule range. If yes then the production rule range test passes.

Block 54: If this block is reached then fire control has failed (logical variable 'pass' is set to false) and the routine returns.

Block 55: This is the first block in the HMS FOV test. Test whether the target was seen by the attacking pilot within the last 10 seconds.

Block 56: Test whether the target is within the HMS FOV. If yes then the HMS test passes.

Block 57: If this block is reached then fire control has failed (logical variable 'pass' is set to false) and the routine returns.

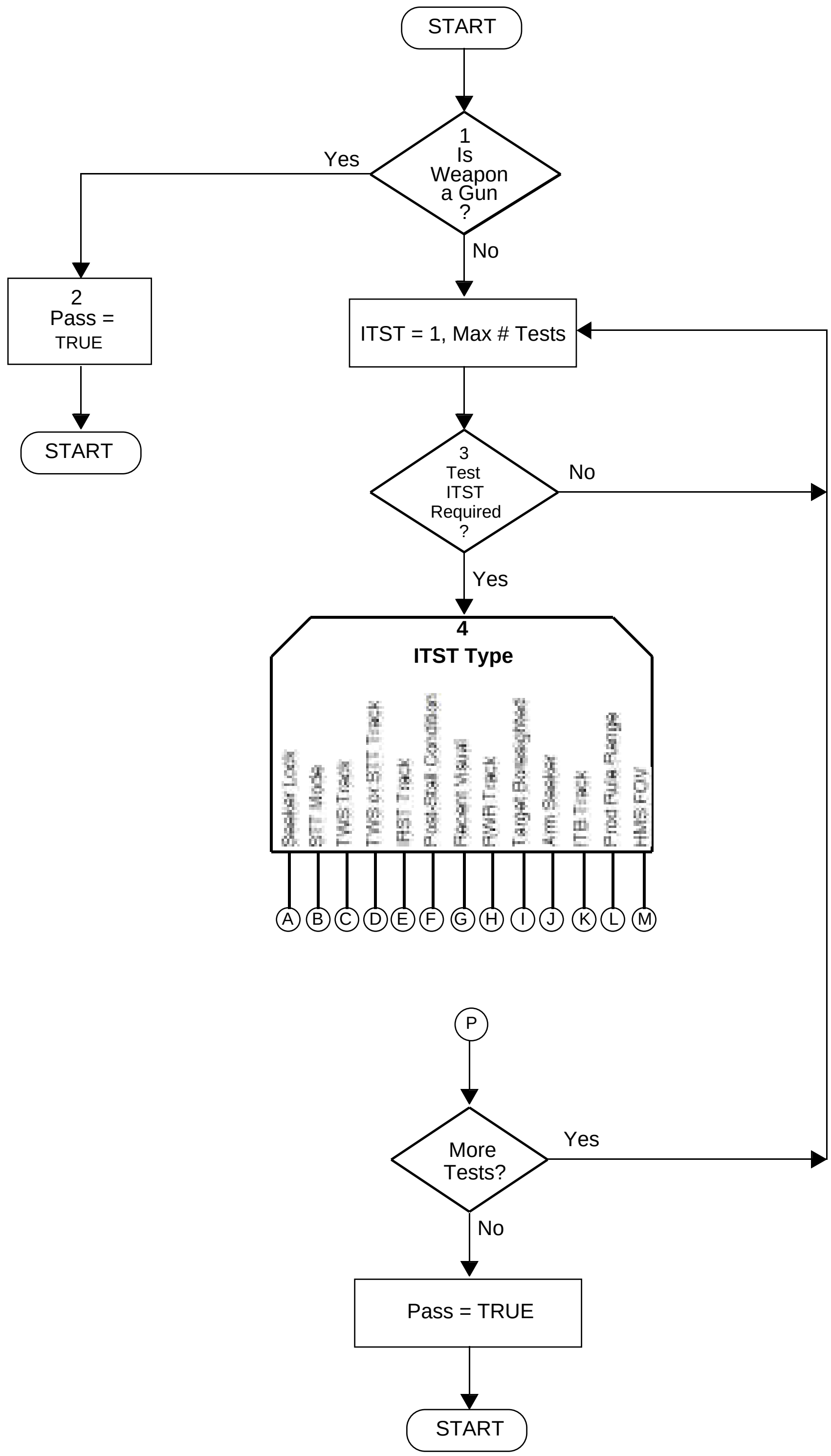

FIGURE 2.24-8. FCTEST Functional Flow Diagram (Page 1 of 9).

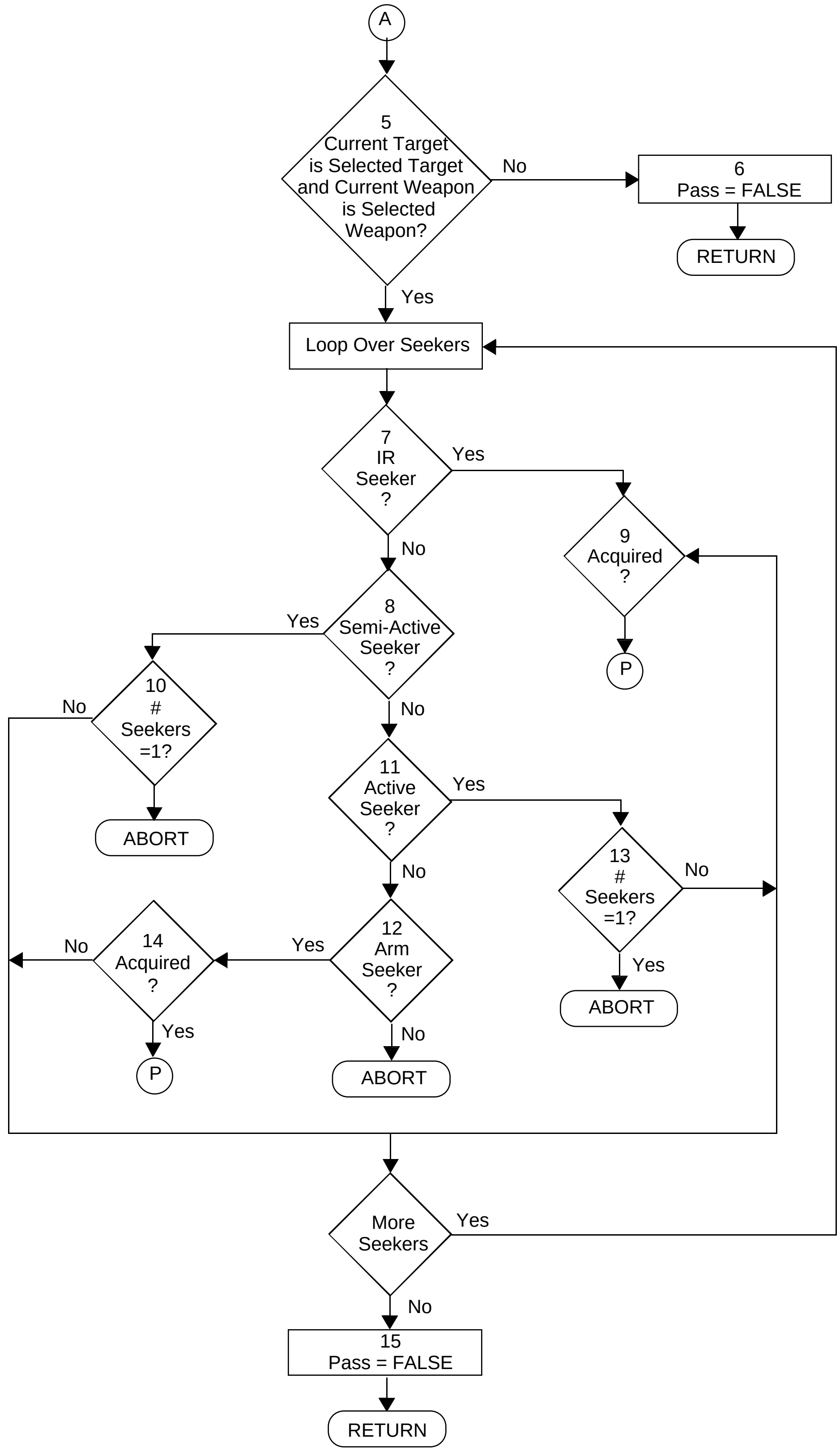

FIGURE 2.24-8. FCTEST Functional Flow Diagram (Page 2 of 9).

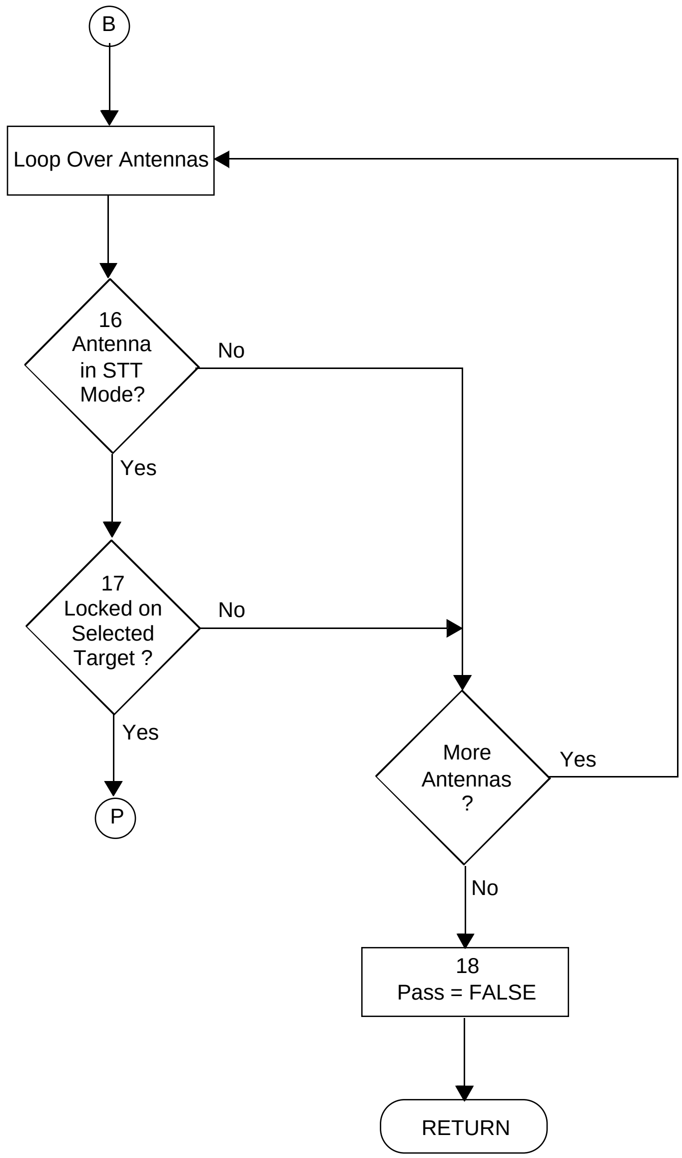

FIGURE 2.24-8. FCTEST Functional Flow Diagram (Page 3 of 9).

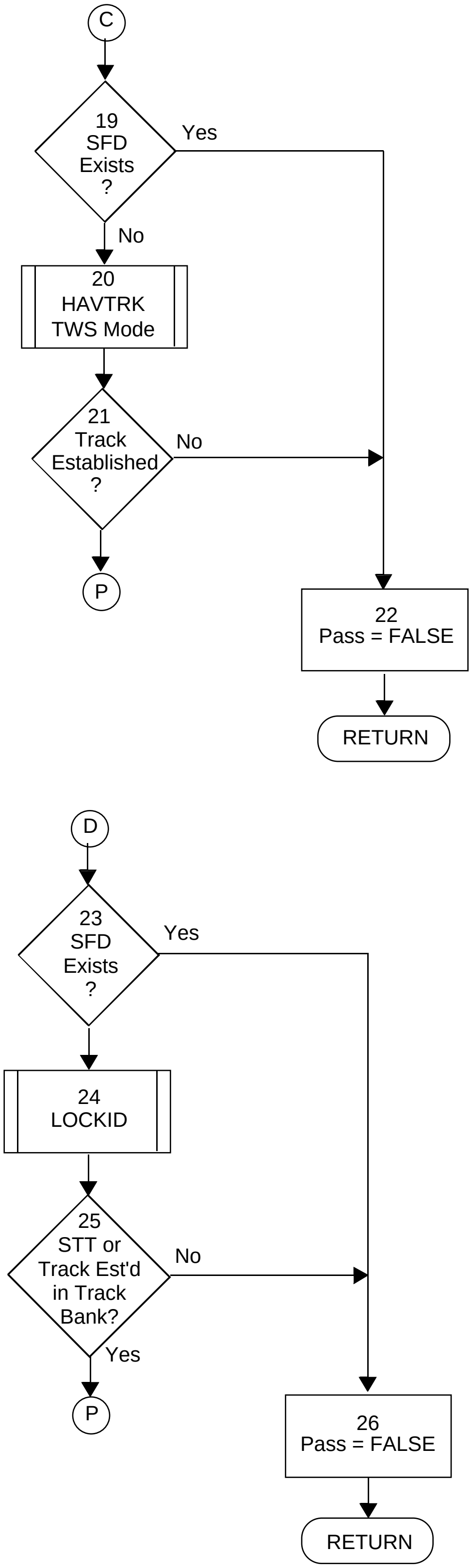

FIGURE 2.24-8. FCTEST Functional Flow Diagram (Page 4 of 9).

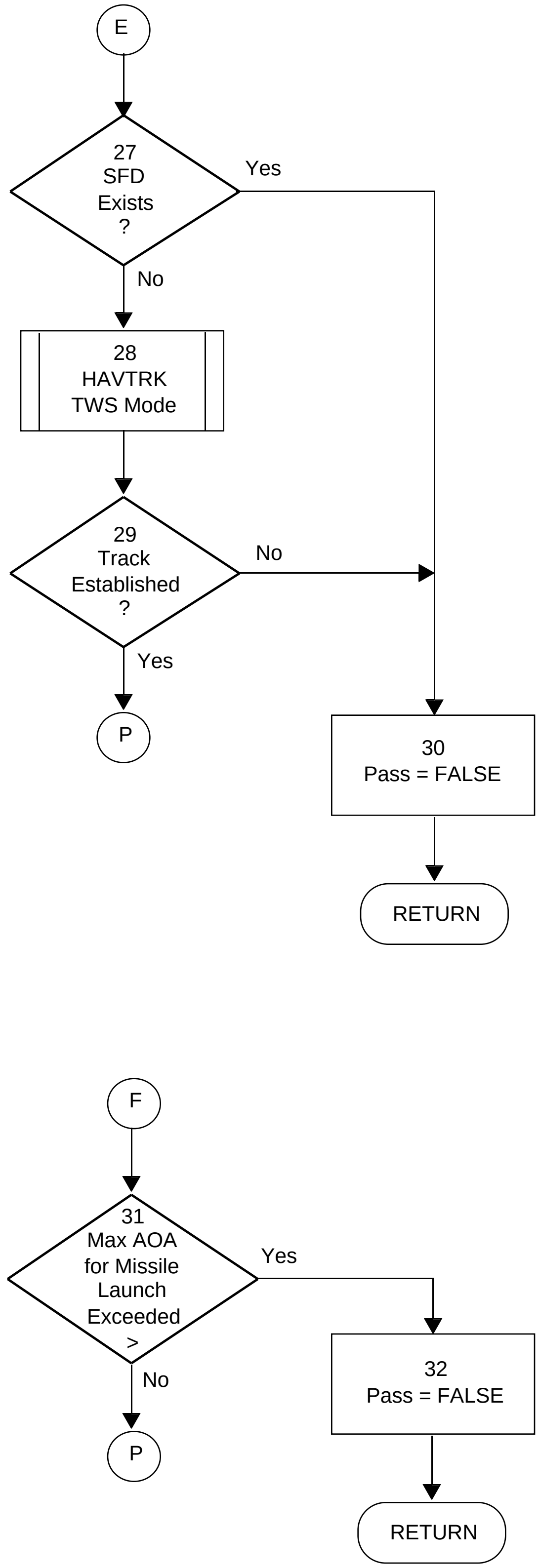

FIGURE 2.24-8. FCTEST Functional Flow Diagram (Page 5 of 9).

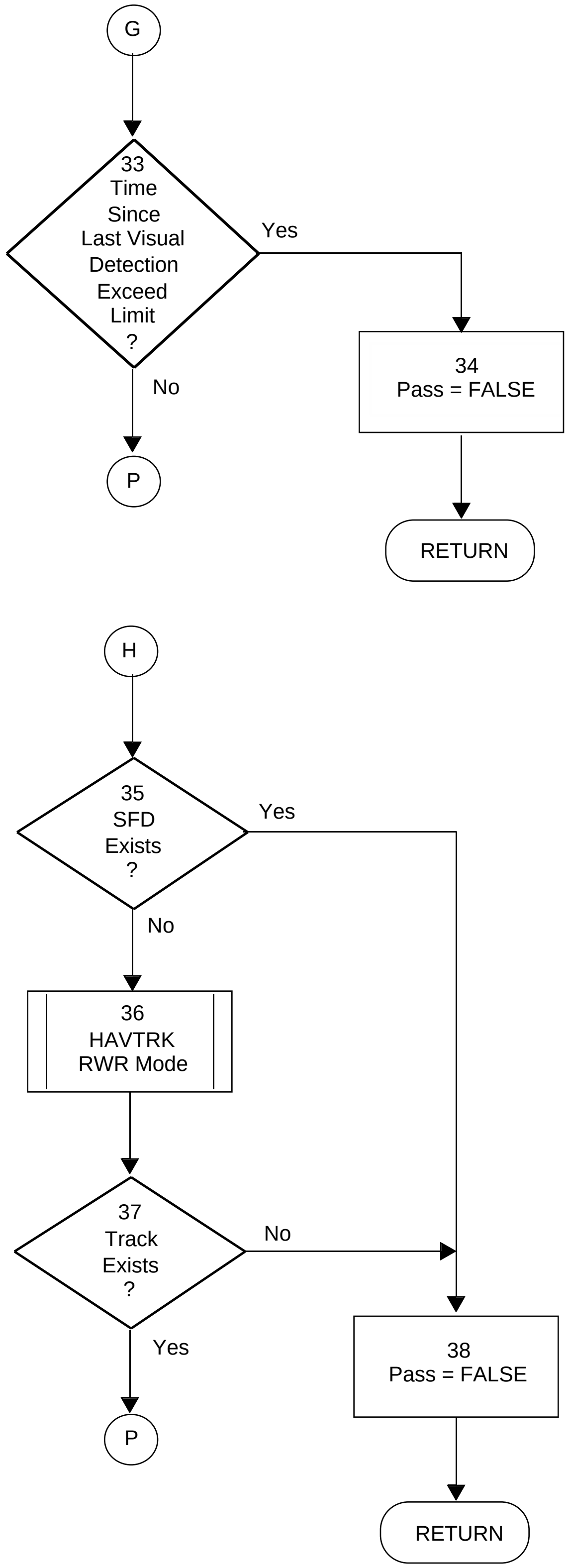

FIGURE 2.24-8. FCTEST Functional Flow Diagram (Page 6 of 9).

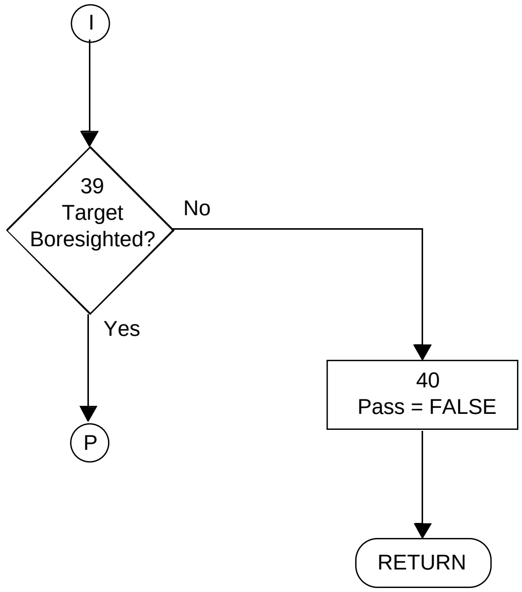

FIGURE 2.24-8. FCTEST Functional Flow Diagram (Page 7 of 9).

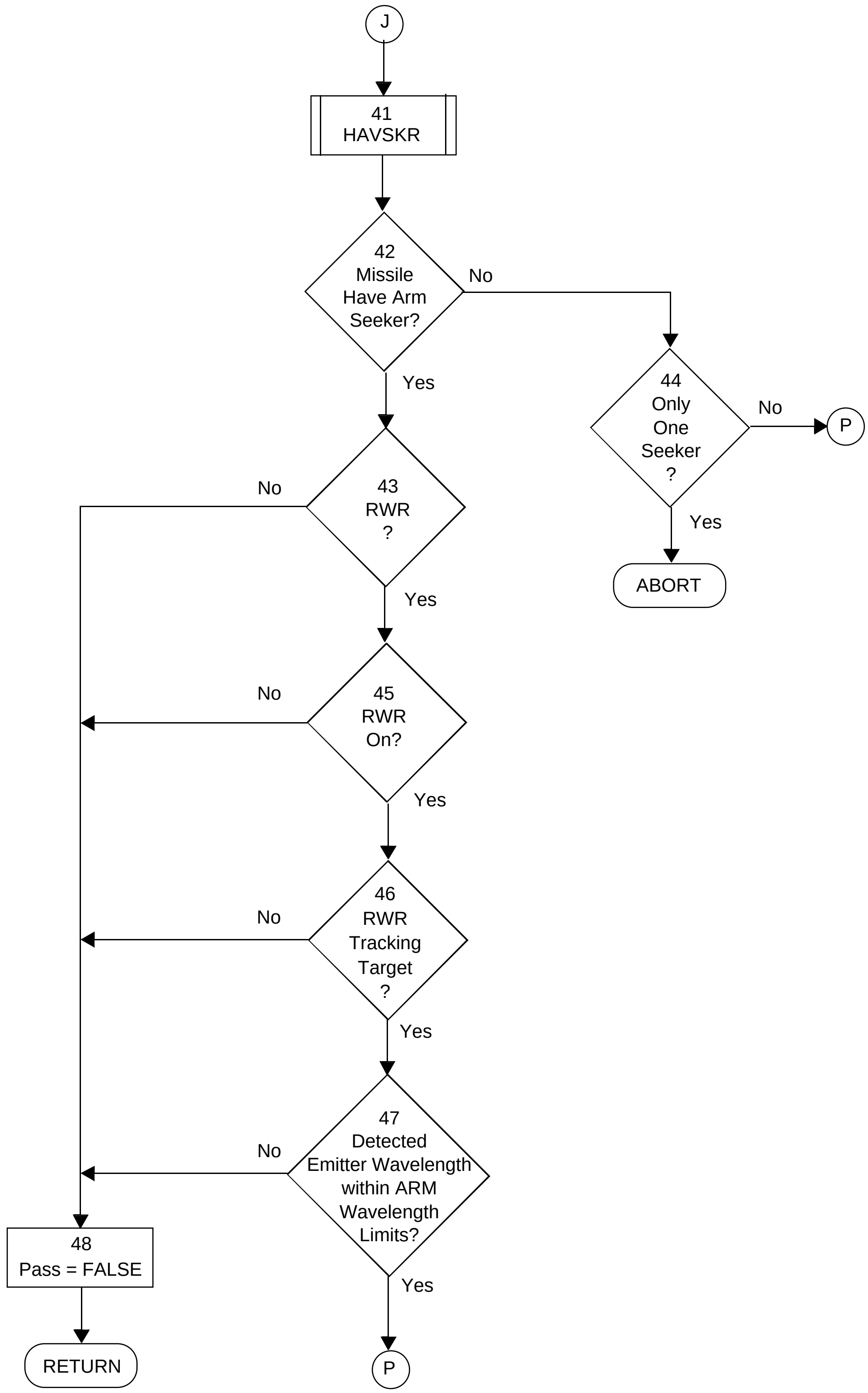

FIGURE 2.24-8. FCTEST Functional Flow Diagram (Page 8 of 9).

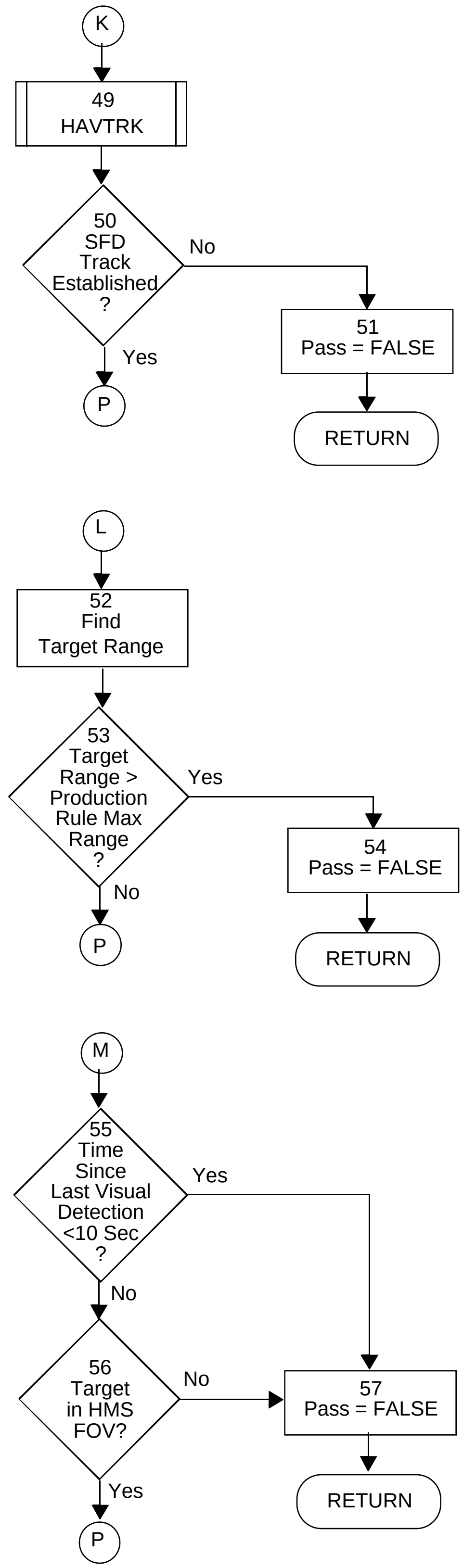

FIGURE 2.24-8. FCTEST Functional Flow Diagram (Page 9 of 9).

Subroutine FCSLCT

Subroutine FCSLCT determines if user-written fire control constraints are satisfied for the current WTP. User written constraints are specified in the **RULES** and **SCNRIO** data files. Figure 2.24-9 is the functional flow diagram that describes the logic used to implement FCSLCT. The blocks are numbered for ease of reference in the following discussion.

Block 1: Subroutine PDEVAL is invoked to initialize the production rule fire control constraint variables.

Block 2: Test whether flight level fire control constraints exist.

Block 3: Subroutine INLSTV retrieves flight level fire control constraint variables.

Block 4: Subroutine BOOLEV determines if the flight level constraints are satisfied. If yes, logical variable ‘slct’ is set to true, otherwise it is false.

Block 5: Subroutine PRSEL is invoked to determine the existence of and evaluate the production rule fire control constraints.

Block 6: Test whether the production rule constraints are satisfied. If no then ‘slct’ remains false and the routine returns.

Block 7: If the production rule constraints were met ‘slct’ is set to true.

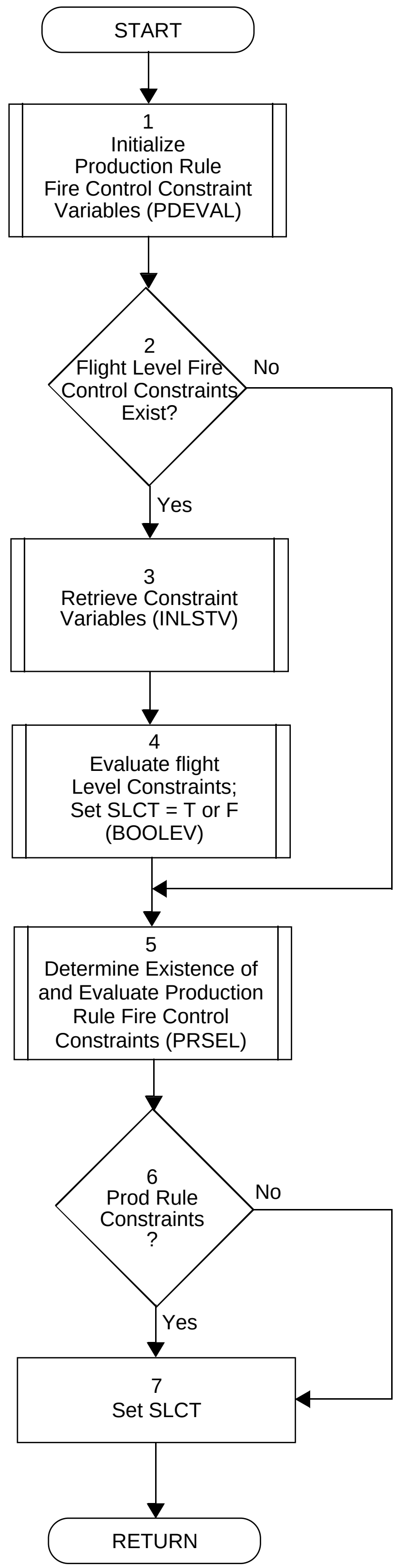

FIGURE 2.24-9. FCSLCT Functional Flow Diagram.

Subroutine EVENVL

Subroutine EVENVL calculates the weapon envelope level for the current WTP. Figure 2.24-10 is the functional flow diagram that describes the logic used to implement EVENVL. The blocks are numbered for ease of reference in the following discussion.

<u>Block 1</u>: A constant velocity projection of the attacker and target aircraft positions is computed.

<u>Block 2</u>: Test whether the target is below ground level. If yes the routine aborts.

<u>Block 3</u>: Test whether the weapon is a gun or missile.

Blocks 4 and 5 are performed for guns

<u>Block 4</u>: Subroutine GUNENV is invoked to determine the gun envelope variables. The current (non-projected) states of the attacker and target are used due to the short time of flight of the bullets.

<u>Block 5</u>: Subroutine ENVLVG is invoked to compute the gun envelope level.

Blocks 6 through 10 are performed for missiles

<u>Block 6</u>: Subroutine MSLENV is invoked with the projected attacker and target positions to compute the missile envelope range limits.

<u>Block 7</u>: Subroutine MSLENV is invoked with the current attacker and target positions to compute the remaining missile envelope parameters.

<u>Block 8</u>: Test whether the attacking platform is a SAM site.

<u>Block 9</u>: If the attacker is a SAM site the steering error is set to zero. This is because SAM launchers are assumed to be able to pivot to point at their targets.

<u>Block 10</u>: Subroutine ENVLVL is invoked to compute the missile envelope level.

Blocks 11 through 13 are in a loop over the number of known friendly aircraft.

<u>Block 11</u>: Test whether the current friendly has targeted the same target as me but after me.

<u>Block 12</u>: Test whether the current friendly is targeting the target that I am considering.

<u>Block 13</u>: If the current friendly is targeting the same target then increase the number of missiles targeted at the current target.

<u>Block 14</u>: Computes the target's survival factor 'facm' as a function of the number of missiles targeted at it.

<u>Block 15</u>: Multiplies the envelope level by 'facm', the value of the utility of engagement, and the probability of kill for the shot.

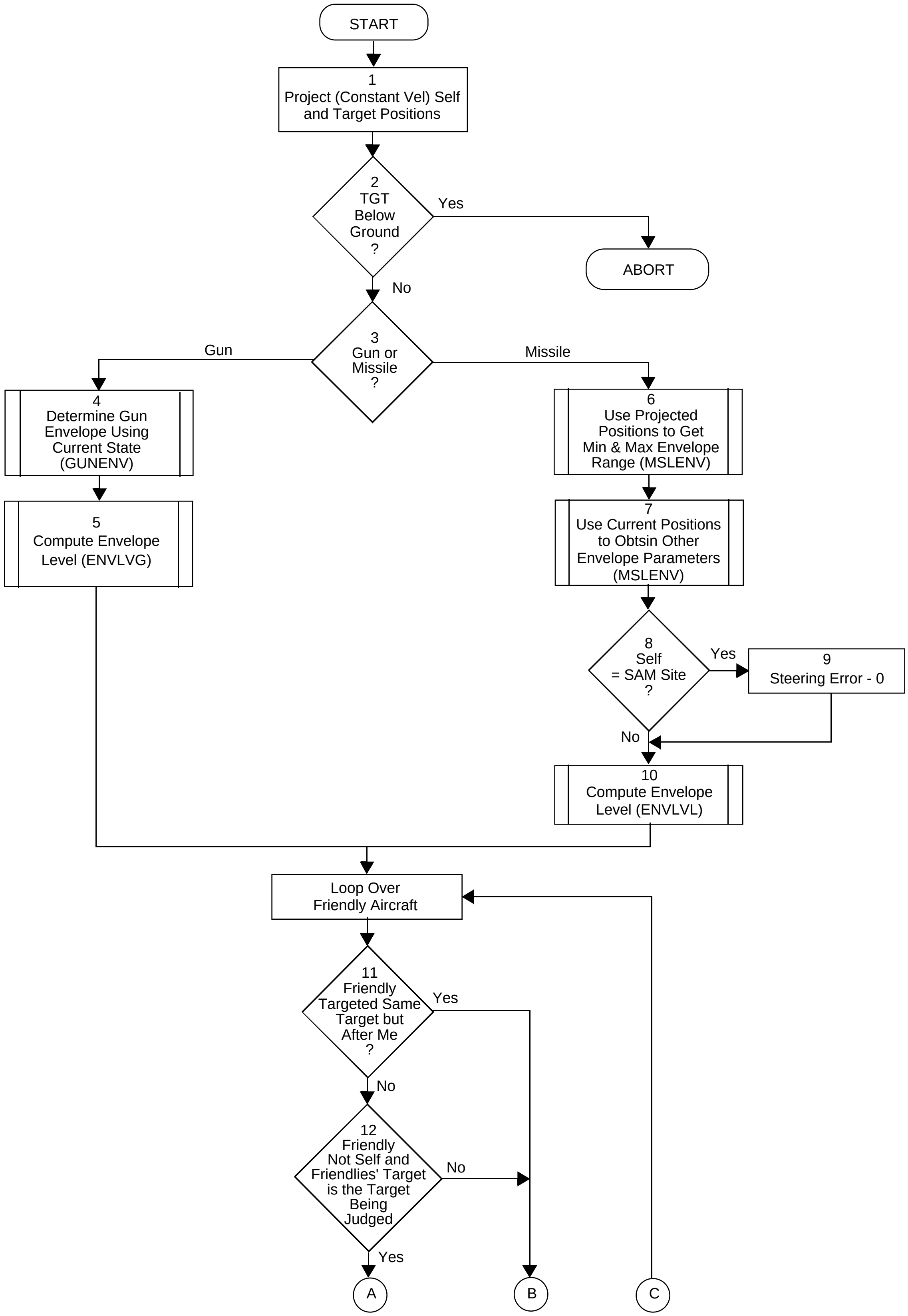

FIGURE 2.24-10. EVENVL Functional Flow Diagram (Page 1 of 2).

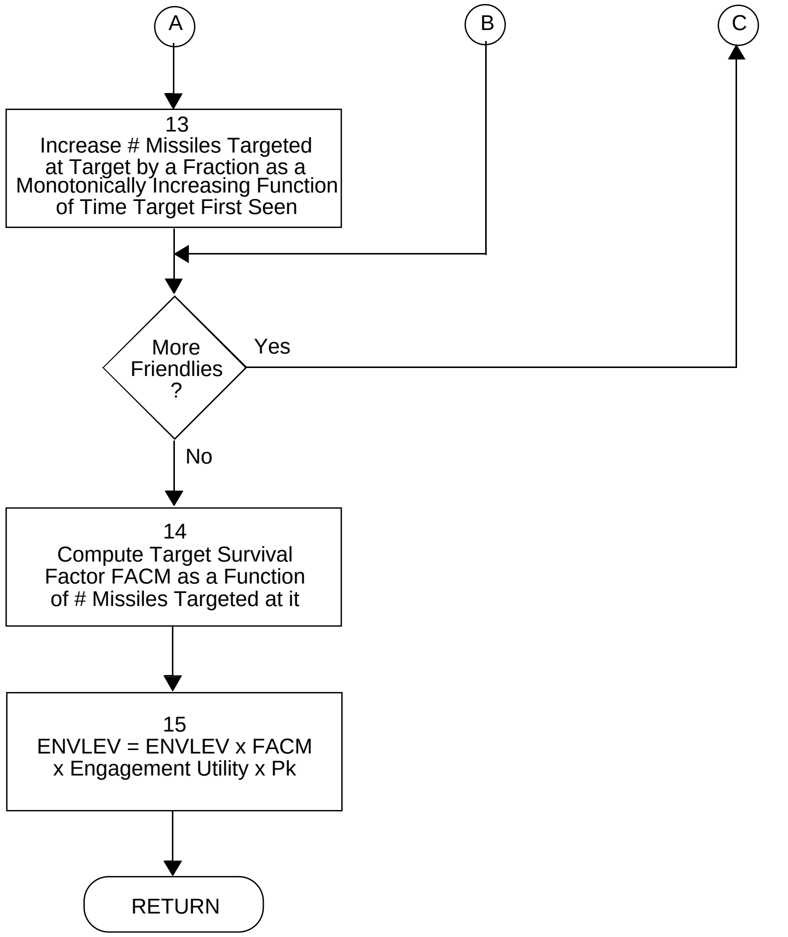

FIGURE 2.24-10. EVENVL Functional Flow Diagram (Page 2 of 2).

Subroutine ADJENV

Subroutine ADJENV adjusts the weapon envelope level for difficulties in acquiring radar lock and costs associated with illumination requirements. Figure 2.24-11 is the functional flow diagram that describes the logic used to implement ADJENV. The blocks are numbered for ease of reference in the following discussion.

<u>Block 1</u>: Test whether the weapon is a non-command guided missile or a gun.

<u>Block 2</u>: Logical variable ‘vislok’ is set to true indicating visual lock is possible if the weapon is a non-command guided IR missile or a gun.

<u>Block 3</u>: Otherwise ‘vislok’ is false indicating visual lock is not possible.

<u>Block 4</u>: Subroutine ANYTRK is invoked to retrieve radar STT mode track status.

<u>Block 5</u>: Test whether the radar is in STT mode against the target.

<u>Block 6</u>: Test whether the weapon is a missile.

<u>Block 7</u>: Test whether any of the missile launch modes require STT.

Block 8: If a launch mode exists that does not require STT then ‘ndlock’ is set to false indicating STT is not required.

Block 9: Test whether STT is required.

Block 10: Test seeker lock is required to launch.

Block 11: Test whether a command guided launch of an IR missile is possible.

Block 12: If IR missile launch without command guidance, then seeker lock is required and ‘ndlock’ is set true.

Block 13: Test whether the missile is a non-SAM site, non-command guided, non-STT IR missile.

Block 14: Test whether the attacker is more than 65 degrees off of the target’s velocity vector.

Block 15: Test whether the attacker is more than 80 degrees off of the target’s velocity vector.

Block 16: Test whether the target is moving into or out of a beam geometry.

Block 17: ‘ndlock’ is set to true indicating STT is required.

Block 18: Test whether STT mode is required.

Block 19: If STT required, compute probability of getting lock, ‘pmiss’.

Block 20: If STT not required, pmiss = 0.

Block 21: Decision on whether the current WTP was the one previously selected and if STT is not required.

Block 22: Compute the time until the shot can be fired (ttgo).

Block 23: Test whether ttgo not equal to zero.

Block 24: Compute range to go ‘rtgo’ if ttgo is not zero.

Block 25: Test whether STT mode is required.

Block 26: Compute ttgo.

Block 27: Compute ttgo for weapon switchology delay.

Block 28: Test whether the current WTP was the one previously selected.

Block 29: Re-compute ttgo for switchology delay.

Block 30: Subroutine ADJTGO to computes the envelope adjustment factor as a function of ttgo.

Block 31: Limit ttgo to 12 seconds maximum.

Block 32: Test whether the missile has a semi-active seeker and will not be launched command guided or is command-guided and the radar is not capable of operating in TWS mode.

Block 33: Compute missile time of flight ‘tofest’ as a function of rtgo.

Block 34: Compute envelope adjustment factor ‘factor’ as a function of tofest.

Block 35: Multiply the envelope value by factor to obtain adjusted envelope factor.

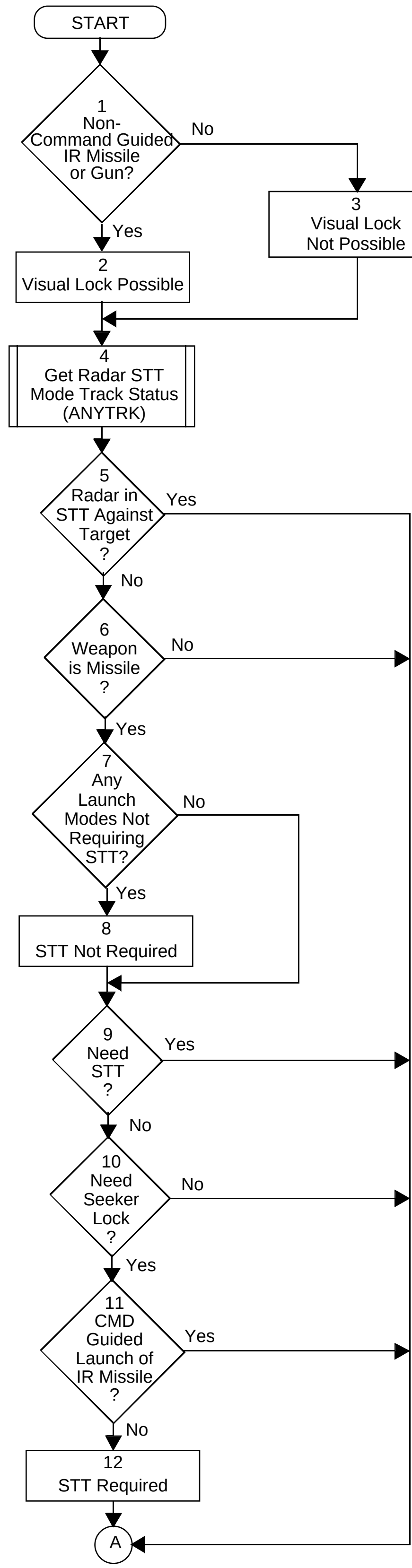

FIGURE 2.24-11. ADJENV Functional Flow Diagram (Page 1 of 3).

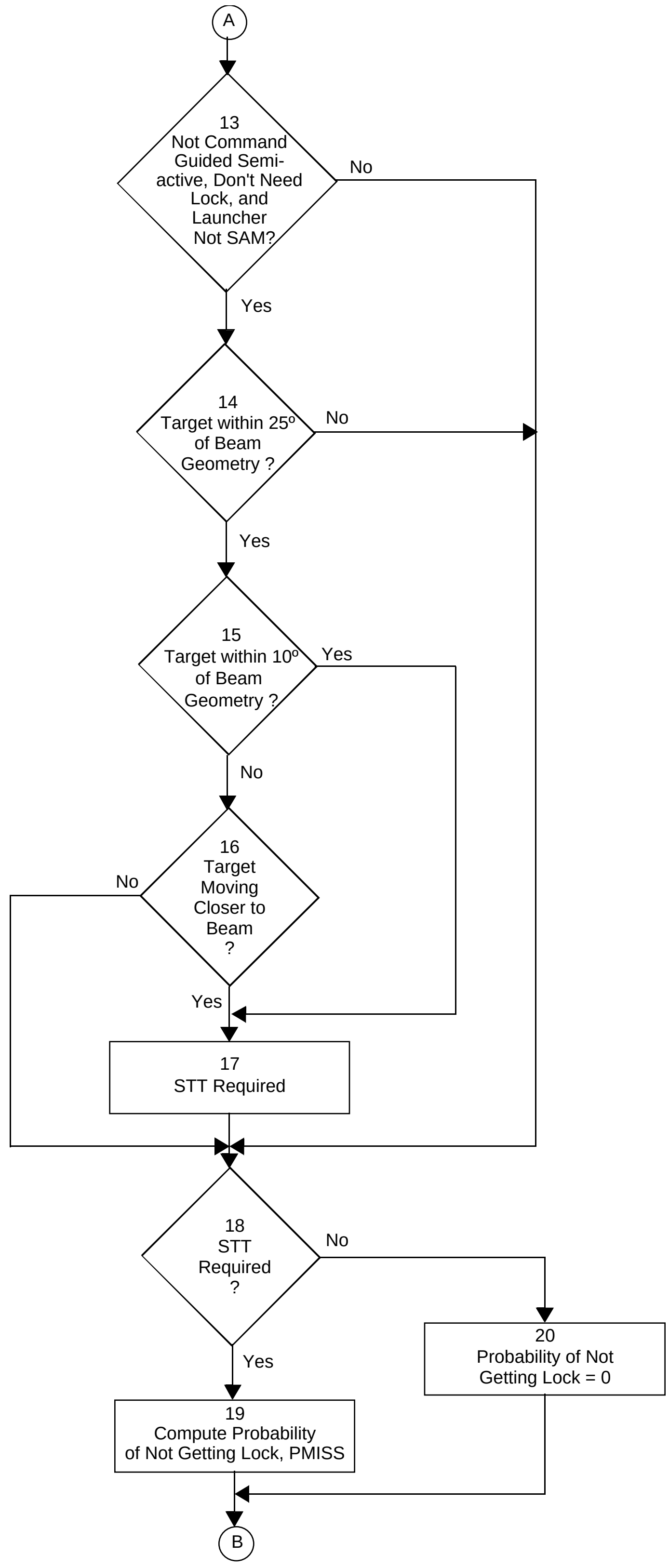

FIGURE 2.24-11. ADJENV Functional Flow Diagram (Page 2 of 3).

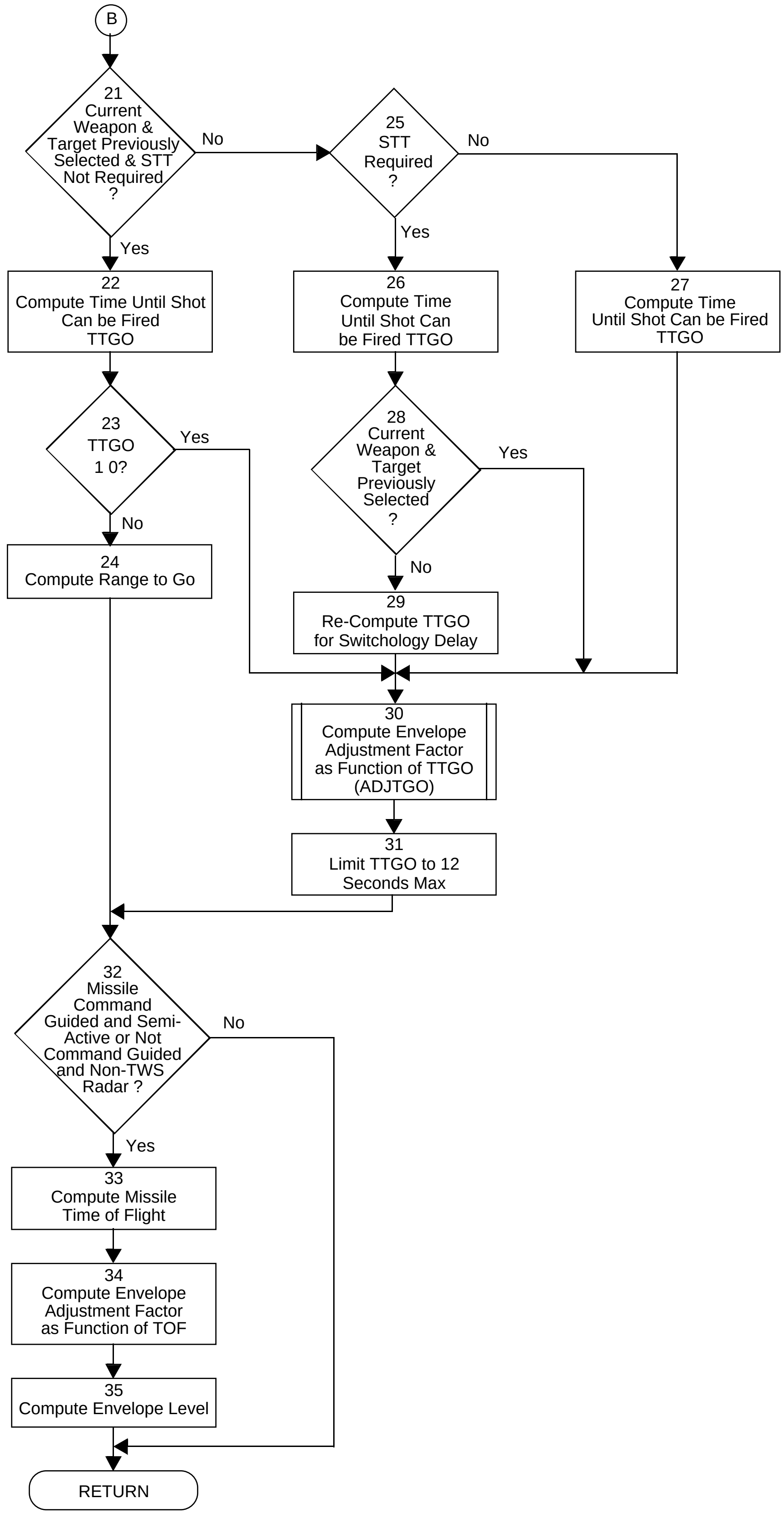

FIGURE 2.24-11. ADJENV Functional Flow Diagram (Page 3 of 3).

Subroutine SET_HLDCOD

Subroutine SET_HLDCOD determines whether or not a target is ineligible because it is too near a friendly or a friendly has a better shot. Figure 2.24-12 is the functional flow diagram that describes the logic used to implement SET_HLDCOD. The blocks are numbered for ease of reference in the following discussion.

Block 1: Test whether any friendly aircraft exist. If no then the routine returns.

Block 2: Subroutine ASSTGT is called to get the target assigned by the flight leader.

Block 3: Subroutine GUNPAR is called to compute aimpoint data (position, etc.).

Block 4: Test whether there is an aimpoint solution.

Block 5: Subroutine GUNPAR is called again to re-compute the aimpoint data with the attacker's speed = target speed + 200 feet per second.

Block 6: Function BORDER is used to determine the shot value 'vshot' as a function of aimpoint range.

Blocks 7 through 19 are in a loop over all known friendlies.

Block 7: Subroutine AVFRAT is invoked to check if the current friendly is in the way of the shot being considered.

Block 8: Test whether the current friendly has a better shot that the attacking aircraft.

Block 9: Test whether the current friendly is believed to be the target of any missiles.

Block 10: Test whether the current friendly's target is also the attacking aircraft's target.

Block 11: Subroutine GUNPAR is invoked to compute aimpoint data for the current friendly.

Block 12: Test whether the current friendly has an aimpoint solution for the target.

Block 13: Use BORDER function to compute the shot value 'valsht' for the current friendly.

Block 14: Test whether the attacking aircraft's aimpoint solution is much better than that of the current friendly.

Block 15: Test whether the current friendly's aimpoint solution is much better than that of the attacking aircraft.

Block 16: Test whether the target is assigned to the current friendly.

Block 17: Test whether the attacking aircraft's aimpoint range is at least 300 feet less than the current friendly's aimpoint range.

Block 18: Test whether the attacking aircraft's aimpoint range is less than the current friendly's aimpoint range + 300 feet.

Block 19: Declare the friendly has a better shot by setting 'hldcode' bit one.

Block 20: Test whether there are more friendlies to loop through.

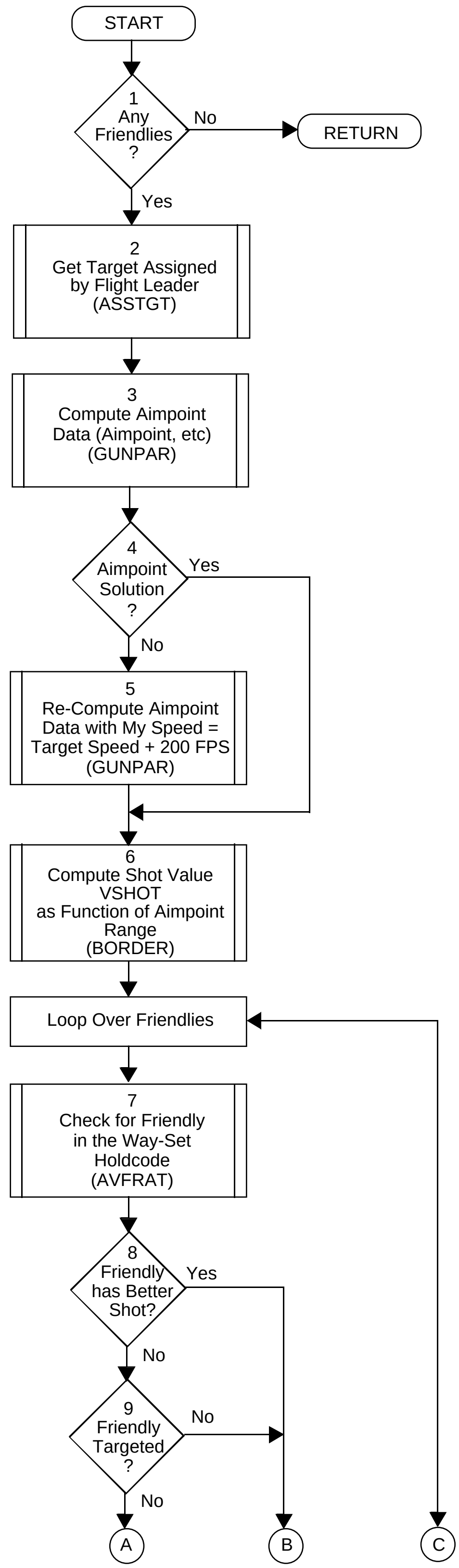

FIGURE 2.24-12. SET_HLDCOD Functional Flow Diagram (Page 1 of 2).

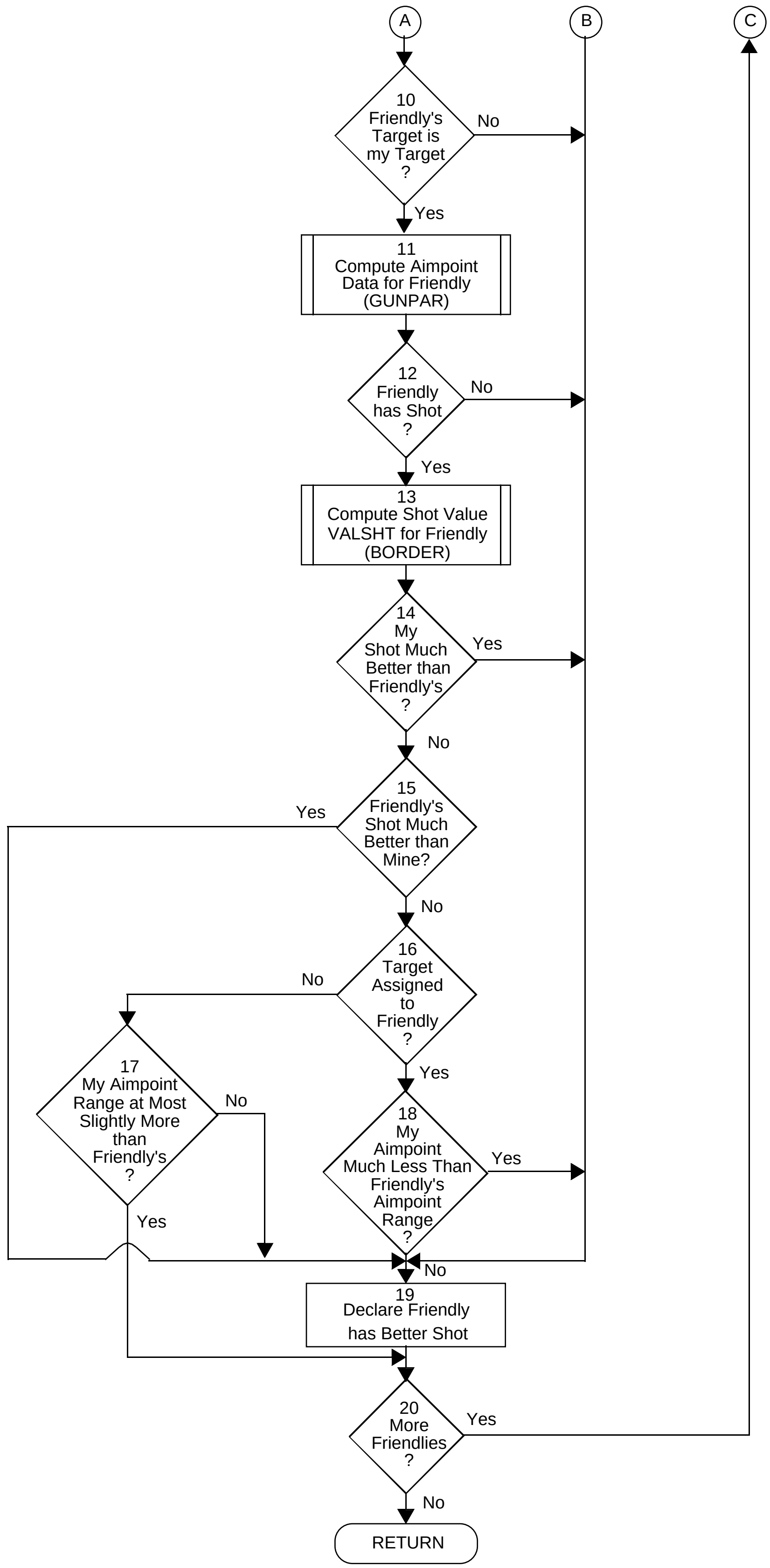

FIGURE 2.24-12. SET_HLDCOD Functional Flow Diagram (Page 2 of 2).

Subroutine SELWPF

Subroutine SELWPF makes the selection of the weapon and target. Figure 2.24-13 is the functional flow diagram that describes the logic used to implement SELWPF. The blocks are numbered for ease of reference in the following discussion.

<u>Block 1</u>: Subroutine VMAX is invoked to search the envelope level list for the highest scoring WTP 'envmax'. In case of a tie between two or more scores the first high score encountered (lowest index) is selected (a property of VMAX).

<u>Block 2</u>: Test whether the highest scoring WTP is above the minimum acceptable value. If not, then none is selected and the routine returns.

<u>Block 3</u>: 'nopick' is set true indicating that no WTP was selected.

<u>Block 4</u>: Subroutine ASSTGT is invoked to determine the target assigned by the flight leader.

<u>Block 5</u>: Add target to target list 'ltgtpp' if not there.

<u>Block 6</u>: Test whether the target is within the weapon range limit.

<u>Block 7</u>: 'nopick' is set true indicating that no WTP was selected.

<u>Block 8</u>: The appropriate pilot posture variables are set for the selected WTP.

<u>Block 9</u>: The time 'timemf' after which the missile can be fired is set.

<u>Block 10</u>: Test whether the current WTP is the previous WTP.

<u>Block 11</u>: Reset time 'tslct' at which the WTP.

<u>Block 12</u>: Subroutine MISLMD is invoked to recompute the WTP envelope level and determine the weapon (missile) mode.

<u>Block 13</u>: Test whether the pilot is interested in firing the weapon.

<u>Block 14</u>: Test whether the pilot previously had no intention of firing or the target is a new one.

<u>Block 15</u>: Reset time 'ppwtim'. This variable records the time at which the pilot began setting switches to fire the selected weapon.

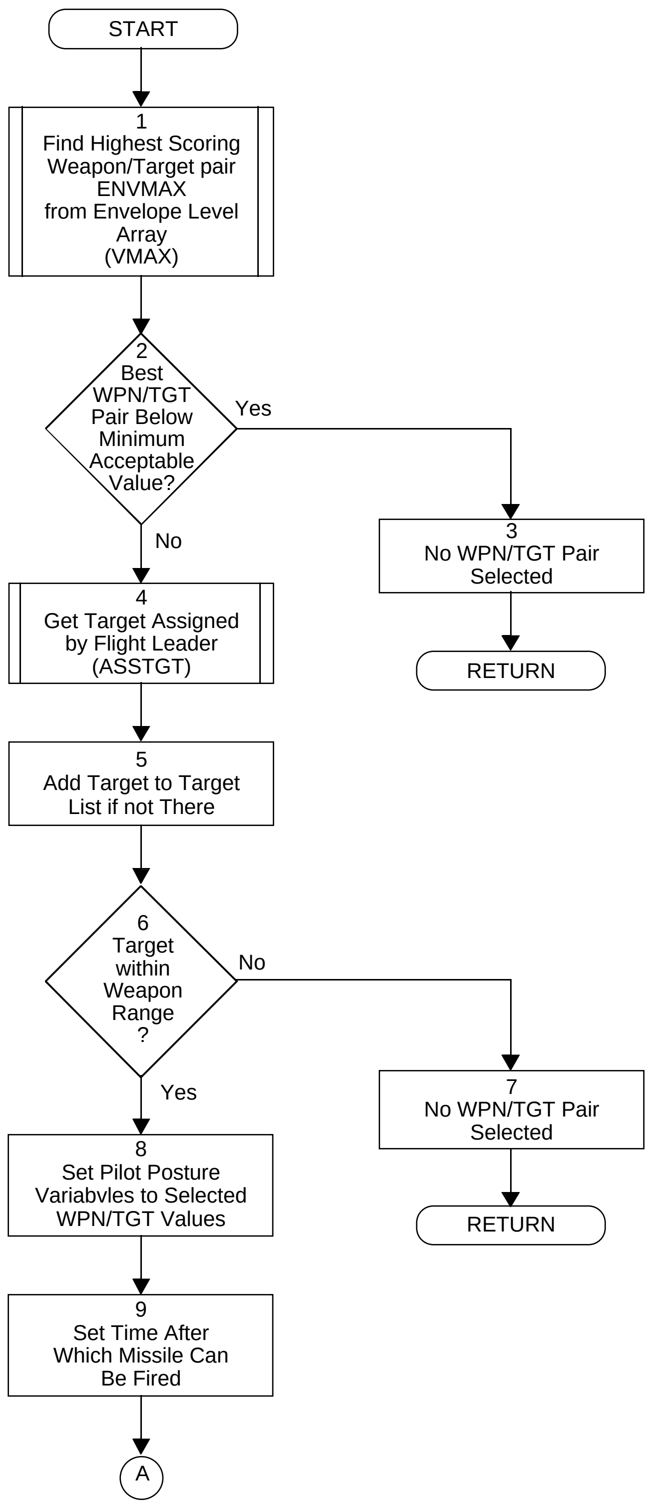

FIGURE 2.24-13. SELWPF Functional Flow Diagram (Page 1 of 2).

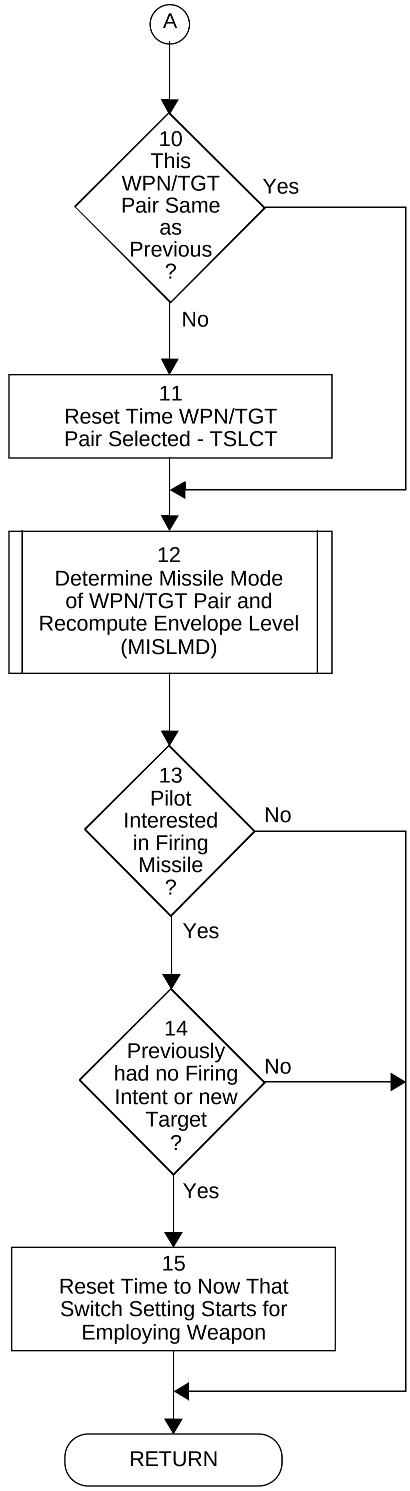

FIGURE 2.24-13. SELWPF Functional Flow Diagram (Page 2 of 2).

Subroutine MISLMD

Subroutine MISLMD determines the weapon (missile) mode of the selected WTP. Possible modes are 0 - no interest in firing, 1 - pilot interested in firing, and 2 - shot is imminent. Figure 2.24-14 is the functional flow diagram that describes the logic used to implement MISLMD. The blocks are numbered for ease of reference in the following discussion.

Block 1: Test whether the WTP is a new one.

Block 2: Subroutine CSKPEN is invoked to remove any seeker acquisition delay penalty. This routine is a holdover from an earlier version of Brawler that imposed a penalty on weapon selection that was a function of seeker acquisition delays. The code that imposed the penalty was removed, but the calls to cskpen that removed the penalty were left alone, as they have no effect on the simulation.

Block 3: Test whether the weapon is a gun or missile.

Block 4: Subroutine GUNENV is invoked to determine the gun firing envelope.

Block 5: Set the maximum steering error, maximum weapon range, and range to target variables.

Block 6: Subroutine ENVLVG is invoked to compute the gun envelope level.

Block 7: Subroutine MSLENV is invoked to determine the missile aimpoint data.

Block 8: Set the maximum steering error, maximum weapon range, and range to target variables.

Block 9: Set mode 'mslmd' to 0.

Block 10: Test whether the attacking entity is a SAM site.

Block 11: Test whether the steering error exceeds the maximum allowed.

Block 12: Set time from envelope 't_aof' to zero if steering error within limit.

Block 13: Test whether t_aof is greater than 10 seconds.

Block 14: Test whether the range to target is greater than the weapon limit.

Block 15: Test whether the range to target is decreasing.

Block 16: Test whether the attacking aircraft will be within weapon range of the target within 10 seconds.

Block 17: Set mode to 1.

Block 18: Test whether the target relationship is unknown and whether it is permissible to fire at unknowns.

Block 19: Test whether the number of missiles designated for the target exceed the maximum allowed.

Block 20: Test whether the attacking entity is a SAM site.

Block 21: Test whether the target will be within the weapon envelope within 2 seconds.

Block 22: Test whether production rules require the attacking pilot to shoot as soon as possible.

Block 23: Set weapon range to absolute maximum allowed by setting the weapon range multiplication factor ‘pp_rfac’ to 1.0.

Block 24: Set weapon range to less than absolute maximum allowed by setting pp_rfac to ppm_rpeak (Rmax2) which is between 0.0 and 1.0. ppm_rpeak indicates the “heart” of the envelope and is set by the user.

Block 25: Test whether range to target is greater than weapon max range.

Block 26: Test whether attacker is closing on target.

Block 27: Test whether the target will be in weapon range within 2 seconds.

Block 28: Test whether the probability of the target being within the maximum weapon range is greater than the minimum probability set by the user. This affects firing decisions in cases when the range to the target is poorly known.

Block 29: Test whether a friendly is in the way of the attacker’s shot.

Block 30: Test whether a friendly has a better shot.

Block 31: Test whether the target is a friendly.

Block 32: Set mode to 2.

Block 33: Subroutine CSKPEN is invoked to remove any seeker acquisition delay penalties.

Block 34: The target is entered into the target list as the only entry on the list.

Block 35: Test whether the mode is zero.

Block 36: Subroutine CSKPEN is invoked to remove any seeker acquisition delay penalties.

Block 37: Test whether the attacking aircraft is additive drag/signature capable.

Block 38: Subroutine ADSTBY is invoked to reset the attacker’s signature/drag status.

Block 39: Test whether production rules override the current mode.

Block 40: Reset mode to production rule value if override occurred.

Block 41: Test whether at least one missile requires illumination.

Block 42: Set mode bit 3 if more than one missile requires illumination - which implies that at least one is already in the air.

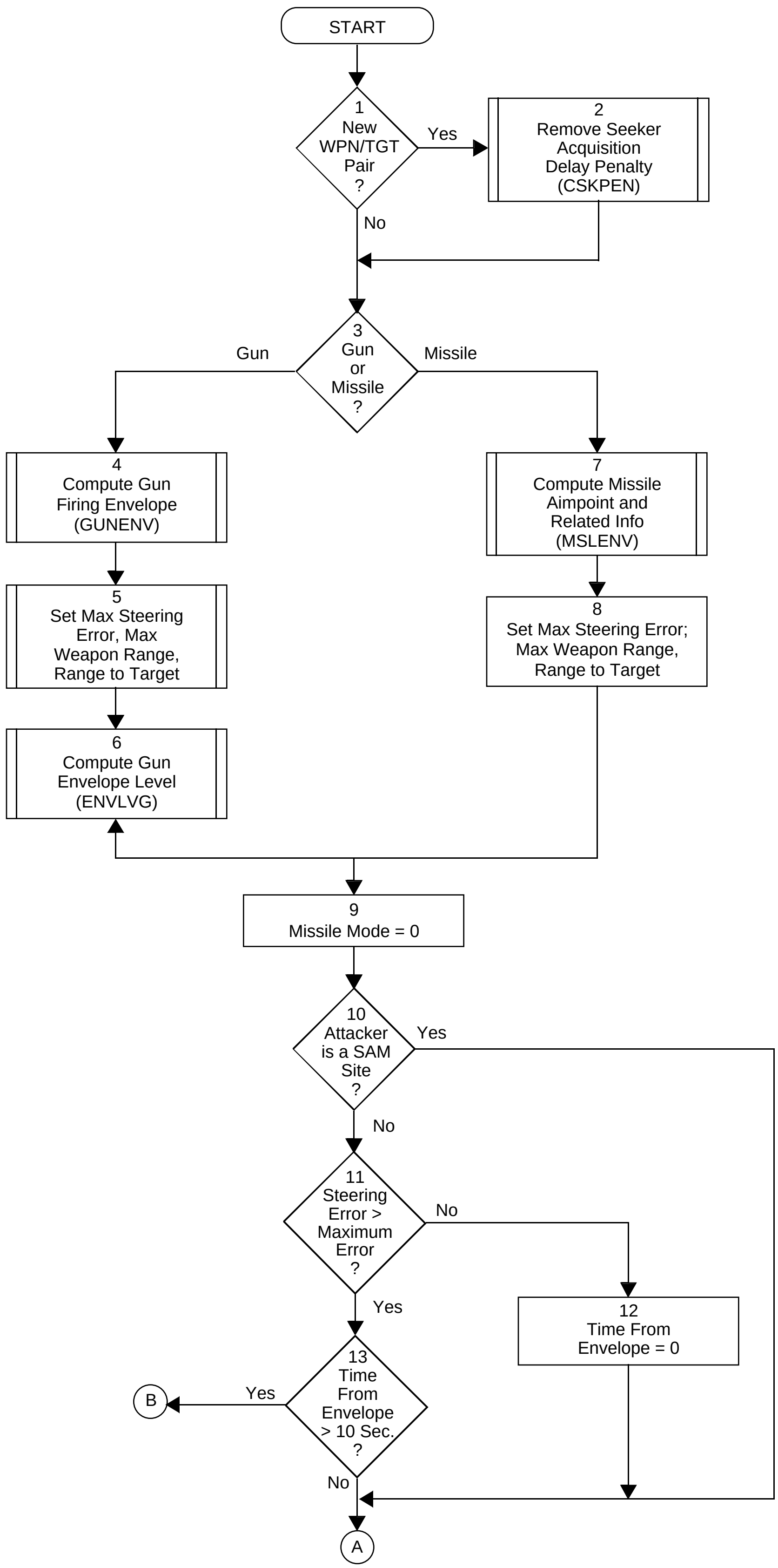

FIGURE 2.24-14. MISLMD Functional Flow Diagram (Page 1 of 5).

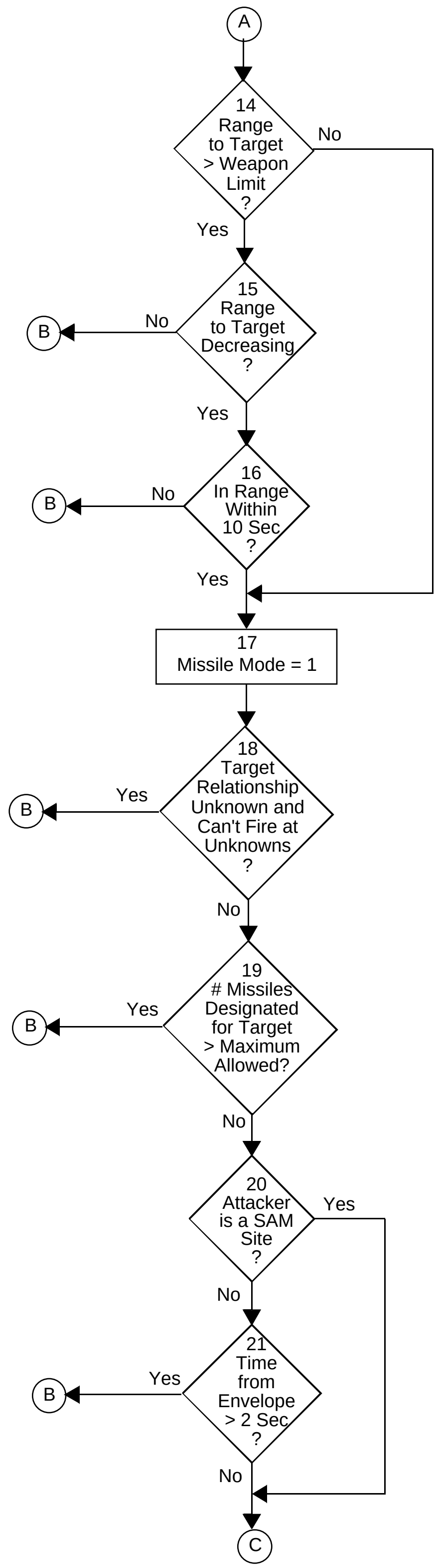

FIGURE 2.24-14. MISLMD Functional Flow Diagram (Page 2 of 5).

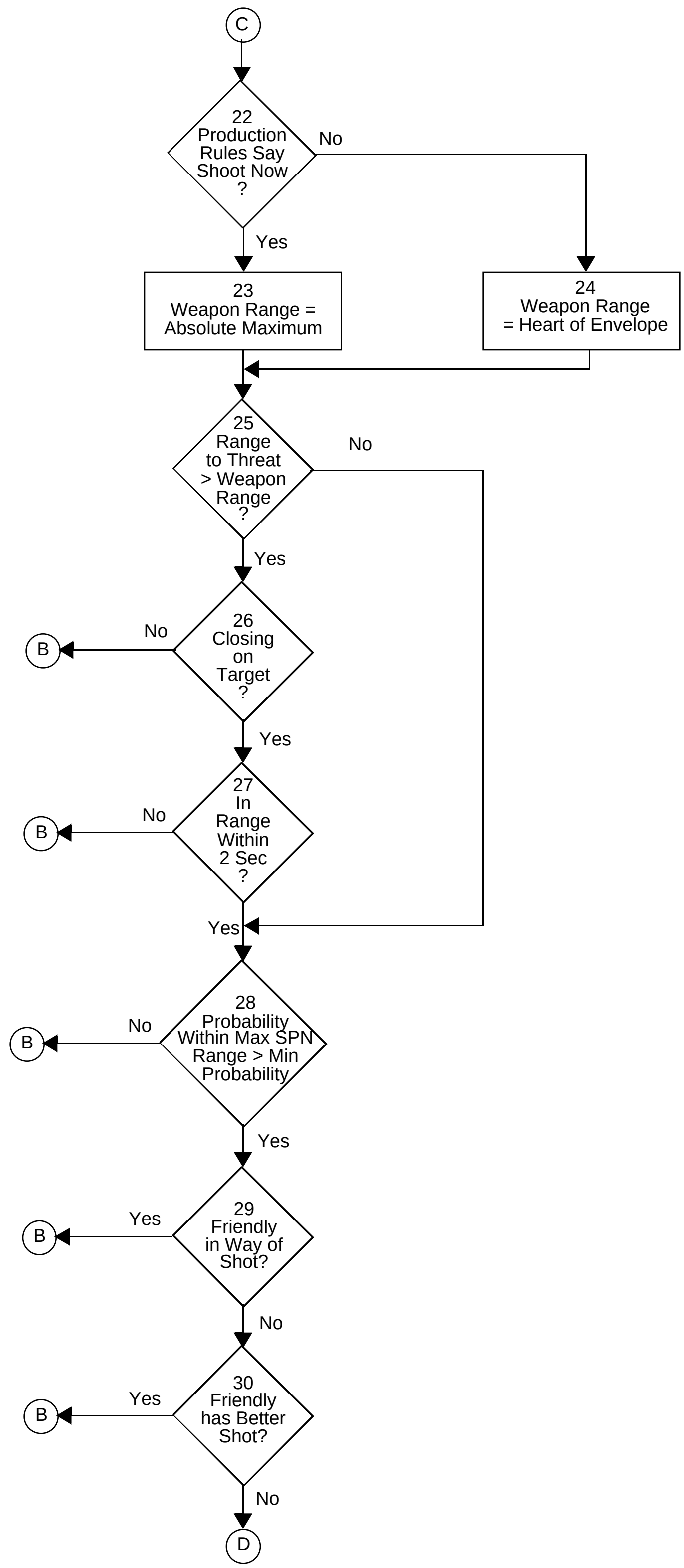

FIGURE 2.24-14. MISLMD Functional Flow Diagram (Page 3 of 5).

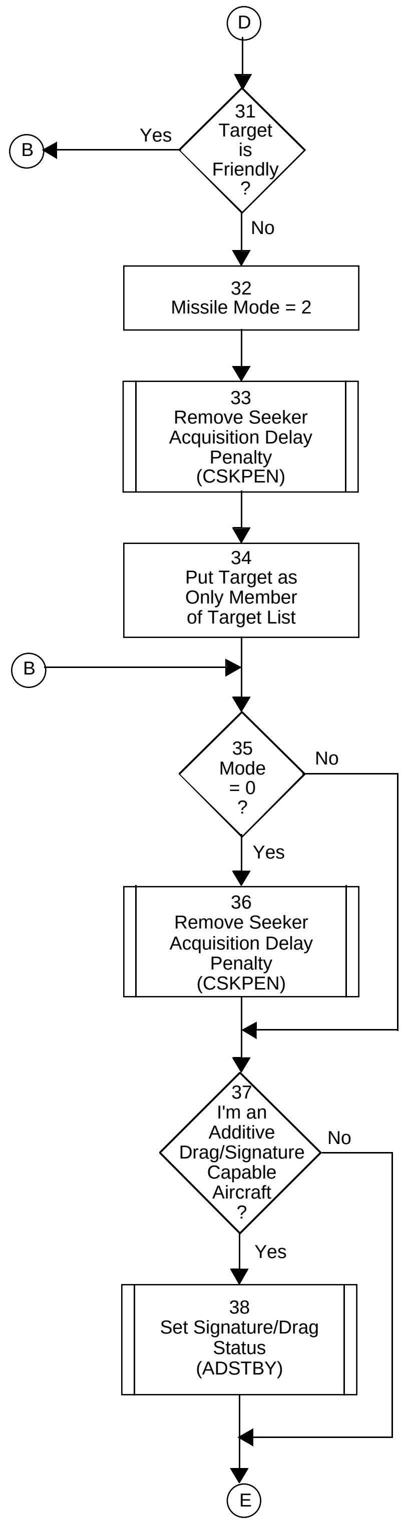

FIGURE 2.24-14. MISLMD Functional Flow Diagram (Page 4 of 5).

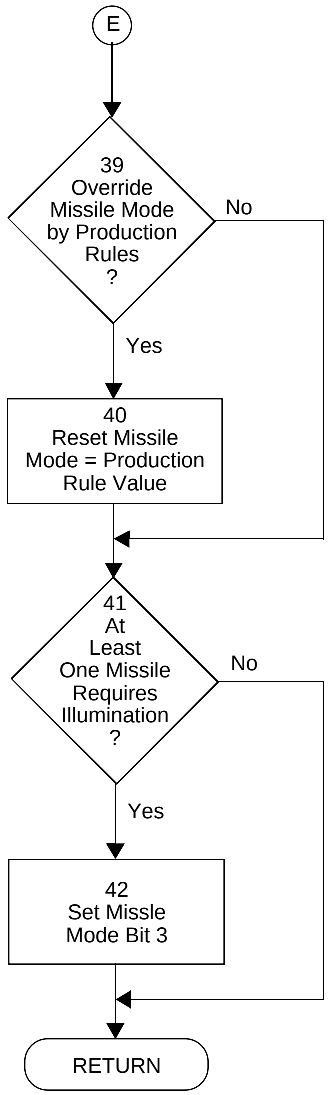

FIGURE 2.24-14. MISLMD Functional Flow Diagram (Page 5 of 5).

TABLE 2.23-1.

System Name	Type
Aircraft	
"ACFT BAC1"	Sample blue aircraft, provided for testing and unclassified demonstrations. Does not represent any real world aircraft type.
"ACFT BAC1RCS"	Same as ACFT BAC1, but with radar cross section data added.
"ACFT BAC1T2"	Same as ACFT BAC1, but with type 2 aero and post-stall data.
"ACFT BAC1TR"	Same as ACFT BAC1, but with thrust reverser.
"ACFT BAC1DD"	Same as ACFT BAC1, but with drag device.
"ACFT SCUD"	Sample initialization data for surface to surface missile.
"ACFT RAC1"	Sample red aircraft, provided for testing and unclassified demonstrations. Does not represent any real world system.
"ACFT RAC1TR"	Same as ACFT RAC1, but with thrust reverser.
"ACFT RAC1DD"	Same as ACFT RAC1, but with drag device.
Expendables	
"CHAFF"	Chaff expendable.
"DECOY"	RF decoy expendable, freely falling.
"FLARE"	IR flare expendable. Illustrates Pk degrade and ballistic effects.
"FLARE3"	IR flare expendable with non-zero ejection velocity.
"FLARE_CEN"	IR flare with IR centroid effect.
"TDECOY"	Towed RF decoy expendable with RF centroid effect and explicit transmitter and receiver gain tables.
"FLARE_CEN3"	IR flare with IR centroid effect and non-zero ejection velocity.
Fire Control Device	
"FCTL_1"	Generic fire control device. Can slave seekers and update missiles from a variety of sources.
"FCTL_2"	Fire control device with longer update and salvo intervals. Can only slave and update with radar tracks.
"FCTL_3"	Like FCTL_1, but can only update missiles with radar or RWR tracks.
"FCTL_4"	Like FCTL_2, but with longer minimum firing interval.
"FCTL_5"	Like FCTL_2, but with shorter minimum firing interval and ability to slave and update from radar or sensor fusion device.
Guns	
"GUN 0"	Generic gun. Uses original gun model.
"GUN 0_MD"	Same as GUN 0, but with additional data to use embedded McDonnel-Douglas gun model.
IFF/NCID	
"EWWS_TEST"	IFF device. Non-cooperative, active emitter type. Victim transponders are TRX_R1, TRX_R2, TEST_RXSP. Illustrates 'NCIDINT' model.
"INFR_RED"	IFF device. Non-cooperative, passive. Illustrates passive non-cooperative 'GENERIC' IFF model.
"PERFIFF"	IFF device. Non-cooperative, active. Illustrates active non-cooperative 'GENERIC' model.
"SAM_SITE"	IFF device. Non-cooperative, active, long range.
"TEST_JEM"	IFF device. Non-cooperative, active. Illustrates 'RADAR' IFF model.
"TEST_RXSP"	IFF device. Cooperative, active. Illustrates 'REALXSP' model.
"TRANSP_1"	IFF device. Cooperative, active. Illustrates active cooperative 'GENERIC' model for blue side.

TABLE 2.23-1. (Contd.)

System Name	Type
"TRX_R1"	IFF device. Cooperative, active. Illustrates active cooperative 'GENERIC' model for red side.
"TRX_R2"	IFF device. Cooperative, active. Illustrates active cooperative 'RADAR' model for red side.
Infra-Red Search and Track	
"IRST 1"	Generic IRST device with laser ranging capability.
"IRST 2"	Generic IRST without laser ranging.
Helmet-Mounted Sight	
"HMSTEST"	Helmet mounted sight.
Missiles	
"MSLG"	Generic semi-active missile. Can be launched command guided, does not require continuous target illumination, seeker performs explicit S/N calculation for acquisition. Other characteristics are the same as MSLR, below.
"MSLI"	Generic IR missile.
"MSLI AC1"	Like MSLI, but with altered drag and missile envelope data.
"MSLI SAM"	Like MSLI, but with larger launch envelope and longer sustain.
"MSLI_TVST"	Like MSLI, but with thrust vs. time engine type and engine reignition.
"MSLI_TVST R"	Like MSLI_TVST, but with different engine reignition data.
"MSLI_LFT"	Like MSLI, but with lofted trajectory with two loft phases.
"MSLI AC2"	IR missile. Less capable than MSLI.
"MSLI6"	Command guided IR missile.
"MSLR"	Semi-active missile.
"MSLR AC2"	Like MSLR, but probability of successful fuzing = 1.0 instead of .78
"MSLR AC1"	Like MSLR AC2, but with less drag and probability of successful launch = .95 instead of .98
"MSLR SAM"	Like MSLR, but longer launch envelope, longer boost phase, shorter sustain phase.
"MSLR_LFT"	Like MSLR, but capable of command guided launch and with a lofted trajectory with two loft phases.
"MSLR/I"	Dual-seeker missile, with semi-active and IR seekers.
Missile Launch Warning Devices	
"MWPERF"	Missile launch warning device with unrestricted field of view, long range.
"MWTEST"	Missile launch warning device with restricted field of view, shorter range.
Radars	
"DFLT_ESA"	Electronically scanned array radar.
"ESA_TEST"	Electronically scanned array radar.
"ESA_2"	Electronically scanned array radar.
"GCI_TEST"	Gimballed radar with 360 degree coverage in azimuth.
"RDR1"	Gimballed radar. High PRF, capable of scan, single target track (STT) modes.
"RDR1 SAM"	Gimballed radar. High PRF, 360 degree coverage, scan/STT modes.
"RDR2"	Gimballed radar. Medium PRF, scan/STT modes.
"RDR2 LKDN"	Like RDR2, but with look down, shoot down capability and uses revised clutter algorithm.
"RDR2 SAM"	Like RDR2, but with 360 degree coverage in azimuth.

TABLE 2.23-1. (Contd.)

System Name	Type
Missile Approach Warning Radar	
“MAWTEST”	Missile approach warning radar.
“MAWPERF”	Missile approach warning radar. More capable than MAWTEST; wider fields of view, more accurate, faster to cue expendables.
Pods	
“ANTI_ESA”	Pod carrying three jammers. Two of type ANTI_ESA_NJ, one of type ANTI_ESA_SJ.
“BLU_ECM_POD”	Pod with two deceptive jammers, one of type BLU_JAM, one of type BL2_JAM.
“POD_NJ”	Pod with one noise jammer of type NOISE.
“POD_PKD”	Pod with one deceptive jammer of type DJAMPK.
“POD_PKN”	Pod with one noise jammer of type NJAMPK.
“POD_SRM_TEST”	Pod with one deceptive jammer of type SRM_TEST.
“RED_ECM_POD”	Pod with two deceptive jammers, one of type RED_JAM, one of type RD2_JAM.
“SMT_POD”	Pod with one deceptive jammer of type SMT_JAM.
“TEST_POD”	Pod with two noise jammers of type JAM1.
“XPOL_POD”	Pod with one deceptive jammer of type XPOL_JAM.
Pylons	
“PYLN_1”	Generic pylon with zero mass, zero additive drag.
“PYLN_3”	Generic pylon with zero mass, zero additive drag.
Radar Warning Receivers	
“RWRPERF”	Sample RWR with 2 fields of view. Capable of sidelobe and missile detection, has ranging capability.
“RWRTEST”	Sample RWR, similar to RWRPERF, but with narrower fields of view.
“RWRTEST2”	Sample RWR, similar to RWRTEST, but is not capable of making sidelobe detections.
Jammers	
“ANTI_ESA_NJ”	Noise jammer effective against radar ESA_TEST and a variety of missiles.
“JAM1”	Generic noise jammer effective against RDR1, RDR2, and a variety of missiles.
“NJAMPK”	Pk-degrade type noise jammer effective against a variety of missiles.
“NOISE”	Generic noise jammer effective against radars RDR1, RDR2, RDR2 LKDN and a variety of missiles.
“ANTI_ESA_SJ”	Deceptive jammer illustrating generic anti-detect and anti-missile effects against radar ESA_TEST.
“BL2_JAM”	Deceptive jammer illustrating generic anti-detect, anti-lock, and anti-missile effects against a variety of radars.
“BLU_JAM”	Deceptive jammer illustrating generic anti-detect, anti-lock, and anti-missile effects against radars RDR1 and RDR2.
“DJAMPK”	Pk-degrade jammer effective against a variety of missiles.
“RD2_JAM”	Deceptive jammer illustrating generic anti-detect, anti-lock, and anti-missile effects against several blue radar systems.
“RED_JAM”	Deceptive jammer illustrating generic anti-detect, anti-lock, and anti-missile effects against several blue radar systems.
“SMT_JAM”	Deceptive jammer illustrating generic anti-detect, anti-lock, and anti-missile effects against radars RDR1 and RDR2.
“SRM_TEST”	Sample swept repeater modulation jammer.
“XPOL_JAM”	Sample cross-pol jammer.

TABLE 2.23-1. (Contd.)

System Name	Type
Tanks	
"TANK"	Generic tank with zero empty mass and zero additive drag.
Sensor Fusion Device	
"SFDPERF"	Generic sensor fusion device.
"SFDTEST"	Generic sensor fusion device similar to SFDPERF, but with more stringent thresholds for track establishment and maintenance.
Situation Awareness Network	
"SANPERF"	Generic situation awareness device.
"SANTEST"	Generic situation awareness device similar to SFDPERF, but with more stringent threshold for track establishment.

2.24.4 Assumptions and Limitations

Weapon choices are limited to air-to-air or ground-to-air missiles and guns. Weapon platforms are limited to SAM sites and aircraft. SAM sites and stand-off jammers cannot be selected as targets. Only the best (highest scoring) WTP is selected per each consciousness event. All possible weapon/target pairs are considered with the following restrictions:

a. Only targets on the hostile target list are considered.

b. No weapon type is considered if no ammunition of that type remains.

c. A weapon type is not considered if firing the weapon would cause the fire control device to exceed its limit on the number of missiles that can be supported with updates.

d. If a weapon requiring radar illumination is in the air and the attacker's radar has no TWS capability, no additional target may be considered for that weapon type or for any other weapon type that requires a radar illumination.

e. A target cannot be considered if the pilot is illuminating other targets and he has neither a track on this target nor a visual sighting.

f. A weapon/target combination cannot be considered unless the weapon's "SELECT" fire control constraints are met for at least one of its launch modes.

g. If flight level fire control constraints are present, the weapon/target combination must pass them.

h. If Production Rule fire control constraints are present, the weapon/target combination must pass them. Note that Production Rule fire control constraints override flight level fire control constraints if both are applied.

i. Surface to air missile (SAM) sites are assumed to be able to turn as required, and so are not subject to steering error constraints.

j. Pilots will not take the possibility of countermeasures into account when selecting a weapon/target pair, even if the target has previously employed countermeasures.

k. Pilots will not "give up" on a weapon/target pair if enough time passes and launch constraints have still not been achieved.

2.24.5 Known Problems or Anomalies

Excessive radio message traffic may delay the receipt of "intent to fire" messages from other aircraft, resulting in multiple attackers firing on the same target. This can often be alleviated by using the fast messages option or by starting engagements with opposing forces outside each other's sensor detection ranges.